

[illegible][illegible]

```
TTTTTTTTTT TTTTTTTTTT YY YY CCCCCCCC HH HH AAAAAA RRRRRRRR 000000  
TTTTTTTTTT TTTTTTTTTT YY YY CCCCCCCC HH HH AAAAAA RRRRRRRR 000000  
    TT      TT      YY YY CC CC          HH HH AA AA RR RR RR RR 00 00  
    TT      TT      YY YY CC CC          HH HH AA AA RR RR RR RR 00 00  
    TT      TT      YY YY CC CC          HH HH AA AA RR RR RR RR 00 00  
    TT      TT      YY YY CC CC          HH HH AA AA RR RR RR RR 00 00  
    TT      TT      YY YY CC CC          HHHHHHHHHH AA AA RRRRRRRR 00 00  
    TT      TT      YY YY CC CC          HHHHHHHHHH AA AA RRRRRRRR 00 00  
    TT      TT      YY YY CC CC          HH HH AAAAAAAAAA RR RR 00 00  
    TT      TT      YY YY CC CC          HH HH AAAAAAAAAA RR RR 00 00  
    TT      TT      YY YY CC CC          HH HH AA AA RR RR RR RR 00 00  
    TT      TT      YY YY CC CC          HH HH AA AA RR RR RR RR 00 00  
    TT      TT      YY YY CC CC          HH HH AA AA RR RR RR RR 00 00  
    TT      TT      YY YY CC CC          HH HH AA AA RR RR RR RR 00 00  
                                     CCCCCCCC CCCCCCCC  
                                     CCCCCCCC CCCCCCCC  
  
LL LL IIIIII SSSSSSSS  
LL LL IIIIII SSSSSSSS  
LL LL II   SS  
LL LL II   SS  
LL LL II   SS  
LL LL II   SS  
LL LL II   SSSSSS  
LL LL II   SSSSSS  
LL LL II   SS  
LL LL II   SS  
LL LL II   SS  
LLLLLLLLLLLL IIIIII SSSSSSSS  
LLLLLLLLLLLL IIIIII SSSSSSSS
```

(2)	237	Declarations
(3)	264	TTY\$GETNEXTCHAR - GET NEXT CHARACTER(S)
(4)	338	WRITING - WRITE state action routine
(5)	415	BURST/STRING - SET UP OUTPUT FOR BURST MODE
(6)	456	FORMAT - FORMAT STRING FOR OUTPUT
(7)	534	OUTPUTANDWAIT - OUTPUT CHARACTER AND WAIT FOR INTERRUPT
(8)	568	FORMAT_CHAR - FORMAT CHARACTER FOR OUTPUT
(9)	621	ADJUST_CURSOR - Increment cursor and set wrap if needed
(10)	651	FORMAT_LOCAL - FORMAT CHARACTER FOR OUTPUT
(11)	697	CHARACTER OUTPUT FORMAT ACTION ROUTINES
(11)	698	BACKSPACE - output a backspace
(12)	724	CARRIAGE - format a carriage return
(13)	803	CTRLZ - output control-Z
(14)	838	ESCAPE - format an escape character
(15)	902	LINEFEED - format a line feed
(16)	950	TAB - Output a tab.
(17)	1012	VTAB, FORM - output a vertical tab, or form feed
(18)	1059	DISPATCHER SERVICE ROUTINES
(19)	1061	BACKSPACING - OUTPUT N BACKSPACES
(20)	1082	CURSOROVRFLOW - insert newline to handle end of line
(21)	1119	EOLSEEN - handle end of line condition
(22)	1159	FILLING - continue outputting fill characters
(23)	1188	MULTIECHOING - Continue outputting multiecho sequence
(24)	1230	SENDLINEFEED - output a line feed
(25)	1255	ESCAPE SEQUENCE PARSING SERVICES
(26)	1273	ESCSYNTAX -- CHECK ESCAPE SEQUENCE SYNTAX
(27)	1347	MOVEREADATA - MOVE CHARACTER FROM TYPEAHEAD TO READ BUFFER
(28)	1481	MOVEREADATA SERVICE ROUTINES
(37)	1736	Specail input character dispatcher
(39)	1812	Post typeahead character action routines
(56)	2335	EDITREAD - READ EDITING STATE
(67)	2542	Read service routines
(75)	2934	Read passall
(76)	3042	READ WITH VERIFICATION
(79)	3434	READ VERIFY ESCAPE TERMINATOR ROUTINE
(80)	3475	End of module


```
0000 1 .TITLE TTYCHARO - Terminal driver character output routine
0000 2 .IDENT 'V04-000'
0000 3
0000 4 *****
0000 5
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23
0000 24 *****
0000 25
0000 26
0000 27 ++
0000 28 Facility:
0000 29
0000 30 VAX/VMS TERMINAL DRIVER
0000 31
0000 32 ABSTRACT:
0000 33
0000 34 THIS MODULE CONTAINS ROUTINES USED FOR THE OUTPUT OF CHARACTERS.
0000 35
0000 36 ***** This module never destroys the contents of R0 *****
0000 37
0000 38 AUTHOR:
0000 39
0000 40 R.HEINEN 11-AUG-1976
0000 41
0000 42 Revision history:
0000 43
0000 44 V03-046 M1R01424 Michael I. Rosenblum 16-Jun-1984
0000 45 Fix bug with read with 0 timeout that creates the typeahead
0000 46 buffer, reported in qar 1424.
0000 47
0000 48 V03-045 M1R0450 Michael I. Rosenblum 27-JUN-1984
0000 49 Make the port bit pc_nocrif set the state skipf
0000 50 Fix bug in escape introducer code that caused it not to
0000 51 recognize csi.
0000 52
0000 53 V03-044 M1R0400 Michael I. Rosenblum 10-Apr-1984
0000 54 Filter null characters from escape sequences.
0000 55
0000 56 V03-043 M1R0390 Michael I. Rosenblum 04-Apr-1984
0000 57 Make the exit string echo thru the tables vector.
```

0000	58	:	
0000	59	:	
0000	60	:	V03-042 MIR0370 Michael I. Rosenblum 20-Mar-1984
0000	61	:	Fix problems with tabs to the left of the current position,
0000	62	:	Lines that wrap in either the prompt or initial string.
0000	63	:	Add port control word modifier to stop free linefeeds
0000	64	:	After typed <CR>.
0000	65	:	
0000	66	:	V03-042 MIR0310 Michael I. Rosenblum 07-Feb-1984
0000	67	:	Fix bugs.
0000	68	:	Noformat reads shouldn't wrap.
0000	69	:	Ignore SKIPCR LF and EDITREAD if we are wrapping.
0000	70	:	Don't change esc to \$ if EDITREAD is set.
0000	71	:	Move to beginning of line should rewrite the whole line
0000	72	:	if the original cursor position was non-zero.
0000	73	:	Read nofltr should obey uppercase rules.
0000	74	:	Passall should not imply termnoecho.
0000	75	:	Fix delete in left justified read verify noecho fields
0000	76	:	to not echo.
0000	77	:	
0000	78	:	V03-041 MIR0300 Michael I. Rosenblum 30-Jan-1984
0000	79	:	Make the uparrow key be command recall.
0000	80	:	
0000	81	:	V03-040 MIR0100 Michael I. Rosenblum 24-Oct-1983
0000	82	:	Fix bug that would cause characters to be echoed in no-echo
0000	83	:	mode.
0000	84	:	
0000	85	:	V03-039 MIR0084 Michael I. Rosenblum 25-Aug-1983
0000	86	:	Fix problem with terminating a right justified numeric
0000	87	:	auto-tab field in read verify.
0000	88	:	
0000	89	:	V03-038 MIR0082 Michael I. Rosenblum 02-Aug-1983
0000	90	:	Fix bug in recall logic that did not force a return before
0000	91	:	the recall was done. Fix bug in delete after wrap logic.
0000	92	:	
0000	93	:	V03-037 MIR0081 Michael I. Rosenblum 01-Aug-1983
0000	94	:	Remove \$brddef call (obsoleted).
0000	95	:	
0000	96	:	V03-036 MIR0080 Michael I. Rosenblum 28-Jul-1983
0000	97	:	Reposition code in the module.
0000	98	:	
0000	99	:	V03-035 MIR0070 Michael I. Rosenblum 13-Jul-1983
0000	100	:	Fix bug that would cause an escape sequence at the beginning
0000	101	:	of the line to return invalid data if deleted.
0000	102	:	Make check in MOVEREADATA for buffer full condition at
0000	103	:	the top of the loop to make sure that the users buffer
0000	104	:	can not be overfilled if the initial string fills the
0000	105	:	data buffer.
0000	106	:	
0000	107	:	V03-034 MIR0053 Michael I. Rosenblum 27-Jun-1983
0000	108	:	Fix bug in ESCAPE_TAB where it would pass the address
0000	109	:	into the ESCINIT routine rather than the character
0000	110	:	
0000	111	:	V03-033 MIR0051 Michael I. Rosenblum 23-Jun-1983
0000	112	:	Fix bug caused when escape sequences are UNUSED.
0000	113	:	Cause simceol to exit to movecursor on hard-copy terminals.
0000	114	:	Make routines that checked CSI and ESC call a common routine.
0000	114	:	Fix bug that would cause writes on non-hardcopy terminals


```
0000 115 : To use the single character code path following the wrap point.
0000 116 :
0000 117 : V03-032 MIR0053 Michael I. Rosenblum 10-May-1983
0000 118 : Fix bug in clear to end of line emulation.
0000 119 :
0000 120 : V03-031 MIR0052 Michael I. Rosenblum 27-May-1983
0000 121 : Add delete character to read verify, make read verify
0000 122 : accept fill characters in the initial string and as
0000 123 : data and display the clear character.
0000 124 : Add software emulation of clear to end of line for non-
0000 125 : ansi terminals.
0000 126 :
0000 127 : V03-030 MIR0050 Michael I. Rosenblum 11-May-1983
0000 128 : Remove code that specail cased broadcasts.
0000 129 :
0000 130 : V03-029 MIR0049 Michael I. Rosenblum 06-May-1983
0000 131 : Add code to handle wrapping correctly.
0000 132 :
0000 133 : V03-028 MIR0041 Michael I. Rosenblum 29-Apr-1983
0000 134 : Fix bug in the new passall code path
0000 135 :
0000 136 : V03-027 MIR0030 Michael I. Rosenblum 30-Mar-1983
0000 137 : Integrate TTRDVFY with the terminal driver, code included
0000 138 : in this module
0000 139 :
0000 140 : V03-026 MIR0029 Michael I. Rosenblum 22-Mar-1983
0000 141 : Add code for insert/overstrike mode.
0000 142 :
0000 143 : V03-025 RKS0025 RICK SPITZ 19-MAR-1983
0000 144 : Change references of UCBSL TT_DEVDP1 TO UCBSL_DEVDEPND2
0000 145 :
0000 146 : V03-024 MIR6026 Michael I. Rosenblum 11-Mar-1983
0000 147 : Change recall to not terminate a small read when a recall
0000 148 : is done from a long recall buffer.
0000 149 :
0000 150 : V03-023 MIR2026 Michael I. Rosenblum 07-Mar-1983
0000 151 : Fix bug in EDITREAD prompt echoing when there is no prompt
0000 152 : to echo.
0000 153 :
0000 154 : V03-022 MIR1026 Michael I. Rosenblum 1-Mar-1983
0000 155 : Add $TRMDEF to the list of required symbol definitions
0000 156 :
0000 157 : V03-021 MIR0026 Michael I. Rosenblum 10-Feb-1983
0000 158 : Add code to impliment input line editing.
0000 159 :
0000 160 : V03-020 MIR0025 Michael I. Rosenblum 18-Feb-1983
0000 161 : Fix bug in carriage control logic that caused a carriage return
0000 162 : in a prompted read to get a free linefeed.
0000 163 :
0000 164 : V03-019 MIR0024 Michael I. Rosenblum 28-Jan-1983
0000 165 : Change code to reflect redefinition of the read packet.
0000 166 :
0000 167 : V03-018 MIR0017 Michael I. Rosenblum 05-Jan-1983
0000 168 : Add powerfail bit to the unit state vector table to remove
0000 169 : powerfail check for each character. Restructure powerfail
0000 170 : code to fall into a common code path in TTYSUB. Change
0000 171 : return status of TTY$GETNEXTCHAR to include a byte value
```

```
0000 172 : in the UCB, this will move the information from the condition
0000 173 : code bits.
0000 174 :
0000 175 : V03-017 MIR0016 Michael I. Rosenblum 29-Dec-1982
0000 176 : Replace time calculation code with TIMSET macro call
0000 177 : This change will allow us to change the time calculation
0000 178 : algorhythm globally if necessary, and will also allow
0000 179 : us to globally turn on and off the time calculations.
0000 180 :
0000 181 : V03-016 MIR0015 Michael I. Rosenblum 22-Dec-1982
0000 182 : Change all calls to the terminal port drivers to refer
0000 183 : to the class jacket routines. Remove DMA code from the
0000 184 : class driver.
0000 185 :
0000 186 : V03-015 MIR0013 Michael I. Rosenblum 16-Dec-1982
0000 187 : Fix up references to new ucb structure
0000 188 :
0000 189 : V03-014 MIR0011 Michael I. Rosenblum 18-Nov-1982
0000 190 : Change MULTIECHO to take a count in UCBSW TT_MULTILEN and
0000 191 : an address in UCBSL TT_MULTI, illiminate the code for
0000 192 : zero terminated multiecho strings.
0000 193 : Change STRTMULTI to take a counted string and convert
0000 194 : it into a length and address.
0000 195 : Add STRTMULTI_1 to take the place of STRTMULTI.
0000 196 : Add code to make Control-R and Control-U look prety on
0000 197 : ANSI crt terminals.
0000 198 : Change CONTROLR to EDITREAD also change the internals of
0000 199 : CONTROLR to use MULTIECHO.
0000 200 : Change the meaning of the CTRLR bit to indicate that the
0000 201 : prompt and data string from a read buffer are being clocked
0000 202 : out by multiecho, this is the only use for CTRLR.
0000 203 : Add SKIPCRLF to indicate that a linefeed following a <CR>
0000 204 : in the beginning of the prompt string is to be skiped.
0000 205 : Remove HOLDSCREEN code.
0000 206 :
0000 207 : V03-013 MIR0012 Michael I. Rosenblum 19-Nov-1982
0000 208 : Fix bug that caused exception when a read was posted
0000 209 : that caused a create of the typeahead buffer.
0000 210 :
0000 211 : V03-012 MIR0010 Michael I. Rosenblum 09-Nov-1982
0000 212 : Move the address of the terminator mask, and the length
0000 213 : of the prompt string from the IRP into the terminal read
0000 214 : packet. Also move the count of the characters in the
0000 215 : buffer from the UCB into the terminal typeahead buffer packet.
0000 216 : Make backspace look like delete (without beeing delete)
0000 217 :
0000 218 : V03-011 RKS0011 RICK SPITZ 19-OCT-1982
0000 219 : INSURE THAT R5 IS PRESERVED WHEN DMA FORK TAKEN
0000 220 :
0000 221 : V03-009 RKS0010 RICK SPITZ 23-SEP-1982
0000 222 :
0000 223 : CHANGE LOGIC IN FORMAT TO INSURE THAT HARDCOPY TERMINALS
0000 224 : SET TO NO WRAP DO NOT OUTPUT DATA BEYOND LINE WIDTH
0000 225 :
0000 226 : CORRECT PROBLEM WITH XON LOGIC TO INSURE THAT OVERFLOW
0000 227 : ERRORS ARE ALWAYS REPORTED
0000 228 :
```


TTYCHARO
V04-000

M 4
- Terminal driver character output routi .16-SEP-1984 02:21:31 VAX/VMS Macro V04-00 Page 5
5-SEP-1984 04:16:19 [TTDRVR.SRC]TTYCHARO.MAR;1 (1)

0000 229 :
0000 230 :
0000 231 :
0000 232 :
0000 233 :
0000 234 :
0000 235 :--

V03-008 MIR0003 Michael I. Rosenblum 12-Aug-1982
Fix bug where multi escape sequences in a TTRDVFY prompt
Would not work.
V03-007 KDM0002 Kathleen D. Morse 28-Jun-1982
Added \$\$\$DEF.


```
0000 237          .SBTTL Declarations
0000 238
0000 239 :
0000 240 : EXTERNAL SYMBOLS
0000 241 :
0000 242
0000 243          $CRBDEF          : DEFINE CRB
0000 244          $IODEF          : DEFINE I/O FUNCTION CODES
0000 245          $IPLDEF        : DEFINE IPL'S
0000 246          $IRPDEF        : DEFINE IRP
0000 247          $PRDEF         : DEFINE PROCESSOR REGISTERS
0000 248          $PRIDEF        : DEFINE PRIORITY CLASSES
0000 249          $RSNDEF        : DEFINE RESOURCE NUMBERS
0000 250          $SSDEF         : DEFINE SYSTEM STATUS CODES
0000 251          $UBADEF        : DEFINE UBA OFFSETS
0000 252          $UCBDEF        : DEFINE UCB
0000 253          $TRMDEF        : TERMINAL ITEMLIST BIT DEFINITIONS
0000 254          $TTYDEF        : DEFINE TERMINAL DRIVER SYMBOLS
0000 255          $TTDEF         : DEFINE TERMINAL CHARACTERISTICS
0000 256          $TT2DEF        : DEFINE EXTENDED TERMINAL CHARS
0000 257          $VECDEF        : DEFINE CRB VECTOR
0000 258          $TTYMACS       : DEFINE TERMINAL MACROS
0000 259          $TTYDEFS       : DEFINE TERMINAL DEFINITIONS
0000 260
0000 261          .PSECT $$$115_DRIVER, LONG : DEFINE NON-PAGED PSECT
0000 262
```

```
0000 264 .SBTTL TTY$GETNEXTCHAR - GET NEXT CHARACTER(S)
0000 265 :++
0000 266 TTY$GETNEXTCHAR - GET NEXT CHARACTER(S)
0000 267 :
0000 268 FUNCTIONAL DESCRIPTION:
0000 269 :
0000 270 THIS ROUTINE RETURNS WITH THE NEXT CHARACTER(S) TO BE OUTPUT ON THE UNIT.
0000 271 :
0000 272 INPUTS:
0000 273 :
0000 274 R5 = UCB ADDRESS
0000 275 :
0000 276 OUTPUTS:
0000 277 :
0000 278 R0 = ALWAYS PRESERVED!!
0000 279 :
0000 280 R3 = 0 AND CC = ZERO - NO CHARACTER TO OUTPUT
0000 281 CHAR AND CC = PLUS - SINGLE CHARACTER TO OUTPUT
0000 282 ADDRESS AND CC = NEG - BURST (R2 = LENGTH)
0000 283 (ADDRESS AND LENGTH ALSO IN UCB)
0000 284 :
0000 285 R5 = UCB ADDRESS
0000 286 UCB$B_TT_OUTTYPE = -1 BIRST
0000 287 ADDRESS IN R3 AND UCB$B_TT_OUTADR
0000 288 LENGTH IN R2 AND UCB$B_TT_OUTLEN
0000 289 0 NO CHARACTER TO OUTPUT
0000 290 1 SINGLE CHARACTER TO OUTPUT IN R3
0000 291 :--
0000 292 :
0000 293 TTY$GETNEXTCHAR::
52 00B8 C5 9E 0000 294 MOVAB UCB$Q_TT_STATE(R5),R2 : ADDRESS STATE OF UNIT
0005 295 :
0005 296 GET CURRENT STATE OF THE OUTPUT
0005 297 :
0005 298 GETNEXTCHAR: : INTERNAL ENTRY
62 00000080 8F D1 0005 299 CMPL #TTY$M_ST_WRITE,(R2) : TRY WRITING GENERAL CASE FIRST
3F 13 000C 300 BEQL WRITING : IF EQL THEN GO
62 00001000 8F D1 000E 301 CMPL #TTY$M_ST_READ,(R2) : TRY READING GENERAL CASE NEXT
03 12 0015 302 BNEQ 10$ : NO, GO TO 10$
050C 31 0017 303 BRW MOVEREADATA : IF EQL THEN GO
001A 304 10$:
54 62 0D 00 EA 001A 305 FFS #0,#TTY$V_ST_READ+1,(R2),R4: GET THE NEXT MOST IMPORTANT OUTPUT STAT
001F 306 CASE R4,TYPE=B,<-- : ENTER THAT STATE'S ROUTINE
001F 307 POWERREST,- : go to the power fail code
001F 308 INTXIT,- : CONTROL S
001F 309 FILLING,- : FILL IN PROGRESS
001F 310 CURSOROVRFLOW,- : CURSOR OVERFLOW
001F 311 SENDLINEFEED,- : LINE FEED TO BE INSERTED
001F 312 BACKSPACING,- : BACKING UP IN PROGRESS
001F 313 MULTIECHOING,- : MULTI ECHO IS PROGRESS
001F 314 WRITING,- : WRITE IN PROGRESS
001F 315 EOLSEEN,- : EOL SEEN
001F 316 EDITREAD,- : READ EDITING OUTPUT STATE
001F 317 RDVERIFY,- : GO TO READ VERIFICATION
001F 318 RECALLING,- : RECALLING THE LAST COMMAND
001F 319 MOVEREADATA> : READ IN PROGRESS
003D 320 :
```

		003D	321	:	EXIT INTERRUPT	
		003D	322	:		
		003D	323	:	INTEXIT:	
010B C5	94	003D	324	:	CLRB UCBSB_TT_OUTTYPE(R5)	: EXIT INTERRUPT
	05	0041	325	:	RSB	: SET NO RETURN CHARACTER
		0042	326	:		: RETURN
		0042	327	:	POWERFAIL DETECTED	
		0042	328	:		
		0042	329	:	POWERREST:	
FFBB'	30	0042	330	:	BSBW RESTART	: CALL THE POWERFAIL RESTART CODE THEN
FFBD	31	0045	331	:	BRW GETNEXTCHAR	: CONTINUE AGAIN
		0048	332	:		
		0048	333	:		
		0048	334	:	INTEXIT1:	
55 BED0		0048	335	:	POPL R5	: EXIT AND RESTORE UCB ADDRESS
F0 11		0048	336	:	BRB INTEXIT	


```
004D 338      .SBTTL WRITING - WRITE state action routine
004D 339
004D 340      ++
004D 341      WRITING - CONTINUE WRITE I/O OUTPUT
004D 342
004D 343      FUNCTIONAL DESCRIPTION:
004D 344
004D 345      THIS ROUTINE IS ENTERED WHEN ALL EXTRA OUTPUT IS COMPLETE
004D 346      AND A WRITE OPERATION IS IN PROGRESS.
004D 347
004D 348      THE NEXT AVAILABLE CHARACTER IS EXTRACTED FROM THE USER BUFFER
004D 349      AND GIVEN TO THE ECHO FORMAT ROUTINES TO OUTPUT CORRECTLY.
004D 350
004D 351      IF CONTROL O HAS STOPPED THE OUTPUT THE OPERATION IS COMPLETED.
004D 352
004D 353      INPUTS:
004D 354
004D 355      R2 = ADDRESS OF THE UNIT STATE VECTOR
004D 356      R5 = UCB ADDRESS
004D 357
004D 358      Implicit inputs:
004D 359
004D 360      UCB$$_TT_WRTBUF - address of write buffer
004D 361
004D 362      OUTPUTS:
004D 363
004D 364      R2 = ADDRESS OF THE TERMINAL STATE VECTOR
004D 365      R3 = CHARACTER TO OUTPUT IF ANY
004D 366      R5 = UCB ADDRESS
004D 367      --
004D 368
004D 369      .ENABLE LSB
004D 370      WRITING:
54 00D4 C5 D0 004D 371      MOVL      UCB$$_TT_WRTBUF(R5),R4      ; Get address of buffer block.
0052 372      IF STATE CTRL0,60$      ; COMPLETE I/O OF CONTROL O
53 1C A4 D0 0056 373      MOVL      TTYS$_WB_NEXT(R4),R3      ; GET ADDRESS OF NEXT CHARACTER
20 A4 53 D1 005A 374      CML      R3,TTYS$_WB_END(R4)      ; DONE?
005E 375      BGEQU     50$      ; IF GEQU THEN YES
0060 376      IF STATE <PASALL,WRTALL>,BURST
0066 377      BRQ      FORMAT      ; FORMAT FOR OUTPUT
0069 378
0069 379      ;
0069 380      ; WRITE I/O DONE
0069 381      ;
0069 382      ;
0069 383      50$:
0069 384      IF NOT_STATE <PASALL,WRTALL>,55$      ; Normal return.
00FC C5 B4 006F 385      CLRW      UCB$$_TT_CURSOR(R5)      ; AT CONCLUSION OF PASSALL/WRITALL
00FF C5 94 0073 386      CLRB      UCB$$_TT_LASTC(R5)      ; STATE IS UNKNOWN
0077 387      55$:
0077 388      MOVW      #SS$ NORMAL,-      ; Load success status code into
0079 389      TTYS$_WB_STATUS(R4)      ; buffer header.
007B 390      BRB      70$      ; CONTINUE
007D 391
007D 392      ;
007D 393      ; CONTROL O TYPED OR CONTROL Y OR C
007D 394      ;
```

```
007D 395
007D 396 60$:
007D 397      MOVW    #SS$ _CONTROL0,-
28 A4 B0 0081 398      TTY$W_WB_STATUS(R4)
0083 399      ; Control-0.
0083 400      ; Load control-0 status code
0083 401 70$:      ; Into buffer header.
0083 402      ; Complete transfer.
0086 403      SUBL3   TTY$L_WB_NEXT(R4),-
53 20 A4 C3 0089 404      TTY$L_WB_END(R4),R3
2A A4 53 A2 008D 405      ; Calculate number of characters
FF70 30 0089 406      SUBW    R3,TTY$W_WB_BCNT(R4)
FF6D 31 008D 407      ; not output to terminal.
0093 408      BSBW    TTY$WRITEDONE
0093 409      BRW     TTY$GETNEXTCHAR
0093 410      ; Adjust transfer count.
0093 411      ; Do I/O done.
0097 412      ; Get the next character.
EA 11 0097 413 WRITE_END:
0099 414      IF STATE CTRL0,60$
0099 415      BRB     70$
0099 416      .DISABLE      LSB
0099 417
```

```
0099 415 .SBTTL BURST/STRING - SET UP OUTPUT FOR BURST MODE
0099 416 :++
0099 417 BURST/STRING -- SET UP OUTPUT FOR BURST MODE
0099 418 :
0099 419 FUNCTIONAL DESCRIPTION:
0099 420 :
0099 421 THIS ROUTINE RETURNS THE ADDRESS AND LENGTH OF THE CURRENT
0099 422 OUTPUT TO THE CALLER. DUETIM IS COMPUTED BASED ON THE LENGTH OF THE
0099 423 OUTPUT. THE CURRENT WRITE POINTERS ARE UPDATED TO REFLECT THE
0099 424 NUMBER OF CHARACTERS INCLUDED IN THE BURST.
0099 425 :
0099 426 INPUTS:
0099 427 R3 = ADDRESS OF NEXT CHARACTER
0099 428 R4 = TWP ADDRESS
0099 429 R5 = UCB ADDRESS
0099 430 :
0099 431 OUTPUTS:
0099 432 :
0099 433 R3 = ADDRESS OF NEXT CHARACTER
0099 434 R5 = UCB ADDRESS
0099 435 UCB$$_TT_OUTADR = ADDRESS OF START OF BURST
0099 436 UCB$$_TT_OUTLEN = LENGHT OF BURST
0099 437 :--
0099 438 BURST:
52 20 A4 53 C3 0099 439 SUBL3 R3,TTY$$_WB_END(R4),R2 ; COMPUTE LENGTH OF RECORD
009E 440 :
009E 441 STR_EXIT:
009E 442 MOVZWL R2,R2 ; CONVERT TO WORD VALUE
1C A4 52 3C 009E 443 ADDL R2,TTY$$_WB_NEXT(R4) ; UPDATE NEXT CHARACTER ADDRESS
0120 C5 52 C0 00A1 444 MOVW R2,UCB$$_TT_OUTLEN(R5) ; SET OUTPUT SIZE
011C C5 53 D0 00AA 445 MOVL R3,UCB$$_TT_OUTADR(R5) ; SET OUTPUT ADDRESS
00AF 446 :
00AF 447 STR_TIMESET:
00AF 448 TIMSET R2,R1,LOCKOUTPUT ; COMPUTE THE TIME PER CHARACTER
00D0 449 : AND SETUP INTERRUPTS
010B C5 01 8E 00D0 450 MNEGB #1,UCB$$_TT_OUTYPE(R5) ; SET NEGATIVE CC TO SIGNAL
00D5 451 : STRING OUTPUT
00D5 452 RSB
00D6 453 :
00D6 454 :
```



```
00D6 456 .SBTTL FORMAT - FORMAT STRING FOR OUTPUT
00D6 457 :++
00D6 458 FORMAT -- FORMAT STRING FOR OUTPUT
00D6 459 FUNCTIONAL DESCRIPTION:
00D6 460
00D6 461 THIS ROUTINE RETURNS THE ADDRESS AND LENGTH OF A PORTION OF THE
00D6 462 CURRENT FORMATTED OUTPUT STRING TO THE CALLER.
00D6 463 IF NO SIGNIFICANT PORTION OF THE STRING CAN BE FOUND, IT
00D6 464 DEFAULTS TO SINGLE CHARACTER FORMAT MODE.
00D6 465 DUETIM IS COMPUTED BASED ON THE LENGTH OF THE
00D6 466 OUTPUT. THE CURRENT WRITE POINTERS ARE UPDATED TO REFLECT THE
00D6 467 NUMBER OF CHARACTERS INCLUDED IN THE BURST.
00D6 468
00D6 469
00D6 470 INPUTS:
00D6 471 R2 = ADDRESS OF UNIT STATE VECTOR
00D6 472 R3 = ADDRESS OF NEXT CHARACTER
00D6 473 R4 = TWP ADDRESS
00D6 474 R5 = UCB ADDRESS
00D6 475
00D6 476 OUTPUTS:
00D6 477
00D6 478 R2 = ADDRESS OF UNIT STATE VECTOR
00D6 479 R3 = ADDRESS OF NEXT CHARACTER
00D6 480 R5 = UCB ADDRESS
00D6 481 UCB$$_TT_OUTADR = ADDRESS OF START OF BURST
00D6 482 UCB$$_TT_OUTLEN = LENGTH OF BURST
00D6 483
00D6 484 OR
00D6 485 R3 = NEXT CHARACTER TO OUTPUT
00D6 486 --
00D6 487 FORMAT:
00D6 488 PUSH R0 ; CSR MUST ALWAYS BE PRESERVED
00D8 489 IF STATE <ESC O>,FORMAT X ; IF ESCAPE IN PROGRESS, SPECIAL
00DC 490 SUBL3 R3,TTYS$_WB_END(R4),R0 ; CALCULATE LENGTH OF STRING
00E1 491 BBS #TTSV_WRAP,UCB$_DEVDEPEND(R5),3$ ; IF WE ARE A NO-WRAP
00E6 492 BBS #TTSV_SCOPE,UCB$_DEVDEPEND(R5),5$ ; SCOPE THEN OUTPUT THE MAXIMUM
00EB 493 ; THAT WE CAN.
00EB 494 3$: SUBW3 #1,UCB$_DEVBUFSIZ(R5),- ; COMPUTE EOL-1
00EF 495 R1
00F0 496 SUBW2 UCB$_TT_CURSOR(R5),R1 ; COMPUTE ROOM TILL EOL-1
00F5 497 BLEQU FORMAT_X ; EOL REACHED
00F7 498
00F7 499 CVTWL R1,R1 ;
00FA 500 CMPL R1,R0 ; OUTPUT SIZE IS LESS OF TWO
00FD 501 BLEQ 10$ ;
00FF 502 5$: MOVL R0,R1 ; ACTUAL IS LESS
0102 503
0102 504 10$:
0102 505 MOVZBL (R3),R0 ; TEST FIRST CHARACTER
0105 506 BITB #<TTYSM_CH_CTRL!TTYSM_CH_SPEC!TTYSM_CH_CTRL2!TTYSM_CH_CTRL3>,-
0108 507 W^TTYSM_CH_CTRL2!TTYSM_CH_CTRL3>,-
010C 508 BNEQ FORMAT_X
010E 509
010E 510 PUSH R1,R2,R3 ; SAVE LENGTH, ADDRESS
0110 511 SCANC R1,(R3) W^TTYSM_CH_CTRL2!TTYSM_CH_CTRL3> ; SKIP ALL NON SPECIALS
0110 511 #<TTYSM_CH_CTRL!TTYSM_CH_SPEC!TTYSM_CH_CTRL2!TTYSM_CH_CTRL3>
0116 512
```

50	20	A4	53	C3	00DC	490
05	44	A5	09	E0	00E1	491
14	44	A5	0C	E0	00E6	492
					00EB	493
	42	A5	01	A3	00EB	494
			51		00EF	495
51	00FC	C5	A2	00F0	496	
		45	1B	00F5	497	
				00F7	498	
	51	51	32	00F7	499	
	50	51	D1	00FA	500	
		03	15	00FD	501	
	51	50	D0	00FF	502	
				0102	503	
				0102	504	
	50	63	9A	0102	505	
	FO	8F	93	0105	506	
0000	CF	40		0108	507	
		2E	12	010C	508	
				010E	509	
		0E	BB	010E	510	
0000	CF	63	51	0110	511	
	FO	8F	2A	0116	512	

```
0118 513
0118 514 20$:
0118 515 POPR #*M<R1,R2,R3>
011A 516 SUBL R0,R1
011D 517 BEQL FORMAT_X
011F 518 ADDW R1,UCBSW_TT_CURSOR(R5)
0124 519
0124 520
0124 521 CLR STATE <NL,SKIPLF,WRAP>
0129 522 MOVE R1,R2
012C 523 POPL R0
012F 524 SUBL3 #1,R2,R1
0133 525 MOVB (R3)[R1],UCBSB_TT_LASTC(R5)
0139 526 BRW STR_EXIT
013C 527
013C 528 FORMAT_X:
013C 529 POPL R0
013F 530 MOVZBL (R3),R3
0142 531 INCL TTY$L_WB_NEXT(R4)
0145 532 BRW FORMAT_CHAR

: TO ALLOW SINGLE BURST OUTPUT
: RESTORE LENGTH , ADDRESS
: COMPUTE NUMBER CHARS FOUND
: YES
: UPDATE FINAL CURSOR
: THIS ASSUMES ALL CHARS
: < SPACE ARE SPECIALS
:
: LENGTH OF STRING TO OUTPUT
: RESTORE
: CALC N-1
: LASTC IS FINAL CHARACTER OUTPUT
: TAKE COMMON STRING EXIT

: RESTORE
: GET CHARACTER
: BUMP NEXT POINTER
: HANDLE AS SINGLE
```

```
014B 534 .SBTTL OUTPUTANDWAIT - OUTPUT CHARACTER AND WAIT FOR INTERRUPT
014B 535 :++
014B 536 : OUTPUTANDWAIT - OUTPUT A CHARACTER AND WAIT INTERRUPT
014B 537 :
014B 538 FUNCTIONAL DESCRIPTION:
014B 539 :
014B 540 : THIS ROUTINE IS USED BY THE OUTPUT INTERRUPT ROUTINES TO RETURN
014B 541 : TO THE DEVICE DEPENDENT CODE TO OUTPUT A CHARACTER. THEIR RETURN
014B 542 : CAUSES THE UNIT TO ENTER A WAIT FOR INTERRUPT STATE.
014B 543 :
014B 544 INPUTS:
014B 545 :
014B 546 : R2 = ADDRESS OF THE UNIT STATE VECTOR
014B 547 : R3 = CHARACTER TO OUTPUT
014B 548 : R5 = UCB ADDRESS
014B 549 :
014B 550 OUTPUTS:
014B 551 :
014B 552 : THE UNIT IS PLACED IN A WAIT FOR INTERRUPT STATE
014B 553 : AND THE CONDITION CODES ARE SET TO PLUS INDICATING
014B 554 : SINGLE CHARACTER IN R3.
014B 555 :
014B 556 : R5 = UCB ADDRESS
014B 557 :--
00FF C5 53 90 014B 558 OUTPUTANDWAIT: ; OUTPUT CHARACTER AND WAIT ENTRY
014B 559 : MOVB R3,UCB$B_TT_LASTC(R5) ; SAVE LAST CHARACTER OUTPUT
014D 560 OUTPUTANDWAIT1:
014D 561 : TIMSET 1,,LOCKOUTPUT ; ENABLE THE TIMER WITH A MINIMUM
0167 562 : ; OF 1 SECOND AND INTERLOCK THE
0167 563 : ; OUTPUT STREAM
010B C5 01 90 0167 564 : MOVB #1,UCB$B_TT_OUTYPE(R5) ; SET THE KIND OF OUTPUT
016C 565 : RSB ; AND RETURN WITH CHARACTER
016D 566
```



```
016D 568 .SBTTL FORMAT_CHAR - FORMAT CHARACTER FOR OUTPUT
016D 569
016D 570 .ENABL LSB
016D 571
016D 572 :++
016D 573 FORMAT_CHAR - FIND PROPER OUTPUT FOR SPECIFIED CHARACTER
016D 574
016D 575 FUNCTIONAL DESCRIPTION:
016D 576
016D 577 THIS ROUTINE TRANSLATES THE SPECIFIED CHARACTER FOR OUTPUT
016D 578 ON THE TARGET UNIT. THE OUTPUT OF THE SEQUENCE IS EITHER
016D 579 THE ORIGINAL CHARACTER OR A STARTUP OF THE PROPER MULTIECHO
016D 580 STRING. CURSOR ADJUSTMENT IS ALSO DONE HERE FOR PRINTING
016D 581 CHARACTERS AND FORM CHARACTERS. IT IS POSSIBLE FOR THE RESULT TO BE NO OUTPUT.
016D 582
016D 583 SEE EACH SPECIAL CHARACTER ROUTINE FOR MORE DETAILS ON ECHOING.
016D 584
016D 585 INPUTS:
016D 586
016D 587 R2 = ADDRESS OF THE UNIT STATE VECTOR
016D 588 R3 = CHARACTER TO TRANSLATE
016D 589 R5 = UCB ADDRESS
016D 590
016D 591 OUTPUTS:
016D 592
016D 593 R2 = ADDRESS OF THE UNIT STATE VECTOR
016D 594 R3 = CHARACTER TO OUTPUT NEXT IF ANY
016D 595 R5 = UCB ADDRESS
016D 596
016D 597 5$: IF NOT STATE <ESC_0>,OUTPUTANDWAIT; BR IF NOT OUTPUT ESC
034F 30 0171 597 BSBW ESCSYNTAX_0 : CHECK CHAR AGAINST ESC SYNTAX
D2 14 0174 598 BGTR OUTPUTANDWAIT : SYNTAX OK, OUTPUT CHARACTER
0176 599 IF NOT STATE <MULTI>,B$ : IF MULTIECHOING AND EDITING
017A 600 IF STATE <EDITING>,10$ : DON'T ZERO CURSOR
00FC C5 B4 017E 601 8$: CLRW UCBSW TT CURSOR(R5) : ZERO CURSOR (COLUMN) POSITION
04 A2 0E E4 0182 602 10$: BBSC #TTYSV_ST_ESC_0,4(R2),- : SEQUENCE ENDED CORRECTLY
7A 0186 603 GOOUT : CLEAR STATE AND BRANCH ALWAYS
0187 604 FORMAT_CHAR:
0187 605 IF STATE <PASALL,ESC_0,WRTALL>,5$ : BR IF FORMAT NOT NEEDED
018F 606 CASE W^TTYSV_TYPET[R3],LIMIT=#1@TTYSV_CH_SPEC,TYPE=B,<-
018F 607 BSPACE,-
018F 608 TAB,-
018F 609 LINEFEED,-
018F 610 VTAB,-
018F 611 FORM,-
018F 612 CARRIAGE,-
018F 613 CTRLZ>
A1 AF 9F 01A4 614 PUSHAB OUTPUTANDWAIT : SET RETURN PC FOR ADJUST CURSOR
01A7 615 : AND FLOW INTO ADJUST_CURSOR
01A7 616
01A7 617 .DSABL LSB
01A7 618
01A7 619
```

Address	Hex	Dec	Label	Instruction	Comment
			01A7 621	.SBTTL	ADJUST_CURSOR - Increment cursor and set wrap if needed
			01A7 622		
			01A7 623	ADJUST_CURSOR:	; ADJUST CURSOR
			01A7 624		
			01A7 625	BITB	#<TTY\$M_CH_CTRL!TTY\$M_CH_SPEC!TTY\$M_CH_CTRL2!TTY\$M_CH_CTRL3>,-
			01AA 626	W^TTY\$A_TYPE[R3]	; TEST FOR SPECIAL
			01AE 627	BNEQ	30\$; NON SPACING CHARACTER
			01B0 628	INCW	UCB\$W_TT_CURSOR(R5) ; ADJUST CURSOR
			01B4 629	CMPL	UCB\$W_TT_CURSOR(R5),UCB\$W_DEVBUFSIZ(R5) ; OVERRUN?
			01BA 630	BGEQU	5\$; YES
			01BC 631		
			01BC 632	25\$:	CLR_STATE - ; Set not at newline
			01BC 633		<NL,SKIPLF,WRAP> ; clear SKIPLF and WRAP.
			01C1 634	26\$:	RSB
			01C2 635		
			01C2 636	5\$:	
			01C2 637	BBS	#TT\$V_WRAP,UCB\$L_DEVDEPEND(R5),10\$; BR IF WRAP ENABLED
			01C7 638	BEQL	25\$; IF EQL THEN AT END OF LINE AND OK
			01C9 639	BBS	#TT\$V_SCOPE, - ; IF SCREEN TERMINAL,CONTINUE
			01CE 640		UCB\$L_DEVDEPEND(R5),25\$;
			01CE 641	TSTL	(SP)+ ; DO NOT RETURN TO BSB TO DROP
			01D0 642	BRW	DROP ; IF HARDCOPY,DROP CHARACTER
			01D3 643	10\$:	SET_STATE CURSOR ; SET CURSOR OVERFLOWED
			01D6 644	BRB	25\$
			01D8 645		
			01D8 646	30\$:	CMPL
			01E0 647	BNEQ	26\$; NO THEN DON'T COUNT IT
			01E2 648	TSTL	(SP)+ ; DO NOT RETURN TO BSB TO DROP
			01E4 649	BRW	ESCAPE ; *** FALL INTO THE ESCAPE CODE

```

01E7 651          .SBTTL FORMAT_LOCAL - FORMAT CHARACTER FOR OUTPUT
01E7 652
01E7 653      :++
01E7 654      :FORMAT_LOCAL - FIND PROPER OUTPUT FOR SPECIFIED CHARACTER
01E7 655
01E7 656      :FUNCTIONAL DESCRIPTION:
01E7 657
01E7 658      :THIS ROUTINE TRANSLATES THE SPECIFIED CHARACTER FOR OUTPUT
01E7 659      :ON THE TARGET UNIT FOR A NOECHO TERMINAL. THE OUTPUT OF THE
01E7 660      :SEQUENCE MAY BE THE STARTUP OF THE PROPER MULTIECHO
01E7 661      :STRING. OTHERWISE, CURSOR ADJUSTMENT IS ALSO DONE HERE FOR PRINTING
01E7 662      :CHARACTERS AND FORM CHARACTERS.
01E7 663      :IT IS POSSIBLE FOR THE RESULT TO BE NO OUTPUT.
01E7 664
01E7 665      :SEE EACH SPECIAL CHARACTER ROUTINE FOR MORE DETAILS ON ECHOING.
01E7 666
01E7 667      :INPUTS:
01E7 668
01E7 669      :R2 = ADDRESS OF THE UNIT STATE VECTOR
01E7 670      :R3 = CHARACTER TO TRANSLATE
01E7 671      :R5 = UCB ADDRESS
01E7 672
01E7 673      :OUTPUTS:
01E7 674
01E7 675      :R2 = ADDRESS OF THE UNIT STATE VECTOR
01E7 676      :R3 = CHARACTER TO OUTPUT NEXT IF ANY
01E7 677      :R5 = UCB ADDRESS
01E7 678      :--
01E7 679
01E7 680      :FORMAT_LOCAL:
01E7 681      :CASE      W*TTYS$A_TYPE[R3],LIMIT=#1@TTYSV_CH_SPEC,TYPE=B,<-
01E7 682      :          BSPACE,-
01E7 683      :          TAB_LOCAL,-
01E7 684      :          LINEFEED,-
01E7 685      :          VTAB,-
01E7 686      :          FORM,-
01E7 687      :          CARRIAGE,-
01E7 688      :          CTRLZ>
01FC 689      :
01FC 690      :SPECIAL FORMATTING WAS NOT NEEDED - SIMPLY INCREMENT CURSOR
01FC 691      :FOR THE LOCALLY-ECHOED CHARACTER AND CONTINUE.
01FC 692      :
01FC 693      :BSBB      ADJUST CURSOR      : ADJUST CURSOR
FE04 31 01FE 694      :BRW      GETNEXTCHAR      : GET ANOTHER CHARACTER
FF44 31 0201 695      :GOOUT: BRW      OUTPUTANDWAIT
  
```



```
0204 697 .SBTTL CHARACTER OUTPUT FORMAT ACTION ROUTINES
0204 698 .SBTTL BACKSPACE - output a backspace
0204 699
0204 700 :++
0204 701 : BACKSPACE - OUTPUT A BACKSPACE
0204 702 :
0204 703 : FUNCTIONAL DESCRIPTION:
0204 704 :
0204 705 : THIS ROUTINE OUTPUTS A BACKSPACE ON A TERMINAL.
0204 706 :
0204 707 : INPUTS:
0204 708 :
0204 709 : R2 = ADDRESS OF THE UNIT STATE VECTOR
0204 710 : R3 = TTY$C_BS
0204 711 : R5 = UCB ADDRESS
0204 712 :
0204 713 : OUTPUTS:
0204 714 :
0204 715 : R2 = ADDRESS OF THE UNIT STATE VECTOR
0204 716 : R3 = TTY$C_BS
0204 717 : R4 = ADDRESS OF THE BS/SPACE/BS STRING IF APPROP.
0204 718 : R5 = UCB ADDRESS
0204 719 :--
0204 720 BSPACE:
00FC C5 B7 0204 721 DECW UCB$W TT_CURSOR(R5) : BACKSPACE
FF3D 31 0208 722 BRW OUTPUTANDWAIT : ADJUST HORIZON
: OUTPUT THE CHARACTER
```

```
0208 724 .SBTTL CARRIAGE - format a carriage return
0208 725
0208 726 :++
0208 727 : CARRIAGE - FORMAT FOR CARRIAGE RETURN
0208 728
0208 729 : FUNCTIONAL DESCRIPTION:
0208 730
0208 731 : THIS ROUTINE SETS UP THE PROPER FILL FOR A CARRIAGE RETURN ON
0208 732 : THE TARGET UNIT.
0208 733
0208 734 : INPUTS:
0208 735
0208 736 : R2 = ADDRESS OF THE UNIT STATE VECTOR
0208 737 : R3 = TTY$C CR
0208 738 : R5 = UCB ADDRESS
0208 739
0208 740 : OUTPUTS:
0208 741
0208 742 : R2 = ADDRESS OF THE UNIT STATE VECTOR
0208 743 : R3 = TTY$C CR
0208 744 : R5 = UCB ADDRESS
0208 745 :--
0208 746 CARRIAGE:
0208 747 CLR_STATE <NL> : SET NOT AT NEW LINE.
020F 748 CROUTPUT:
020F 749 IF_STATE - : If in a write state, branch
020F 750 WRITE,115$ : forward.
0213 751 IF_NOT_STATE - : If not in a read state, also
0213 752 READ,115$ : branch forward.
0217 753 IF_STATE <MULTI,EDITREAD>,130$ : not a character echo or wrap
021E 754 MOVL UCB$S_SVAPTE(R5),R1 : GET THE PACKET ADDRESS
0222 755 MOVZWL TTY$W_RB_LINOFF(R1),R4 : GET THE OFFSET TO THIS CHARACTER
0226 756 MOVAB @TTY$C_RB_LIN(R1)[R4],TTY$S_RB_LIN(R1); AND MAKE THIS THE NEW
022C 757 CLRW TTY$W_RB_LINOFF(R1) : LINE-BEGINNING
022F 758 CLRW TTY$W_RB_LINREST(R1) : JUST MAKE SURE WE ARE AT THE END
0232 759 BBC #TTY$V_PC_NOCRLF,UCB$W_T_PRTCTL(R5),110$;SO JUST ECHO THE CHARACTER
0238 760 SET_STATE SKIPLF : NO FREE LINEFEED
023C 761 BRB 115$ : ELSE DON'T DO THE LINEFEED
023E 762
023E 763 130$: IF_NOT_STATE CTRLR,110$ :
0242 764
0242 765 : THE FOLLOWING CODE WILL INSERT A LINEFEED AFTER ANY CARRIAGE RETURN
0242 766 : THAT IS THE LAST CHARACTER IN THE READ PROMPT STRING OR IN THE READ BUFFER
0242 767 : ITSELF. ANY <CR> CHARACTERS IN THE PROMPT STRING PRIOR TO THE LAST CHARACTER
0242 768 : ARE NOT TOUCHED.
0242 769 :
0242 770 MOVL UCB$S_SVAPTE(R5),R1 : GET THE READ PACKET ADDRESS
0246 771 MOVZWL TTY$W_RB_PRLLEN(R1),R4 : GET THE PROMPT LENGTH
024A 772 MOVAB TTY$A_RB_PRL(R1)[R4],R4 : AND THIS IS THE ADDRESS
024F 773 CMPL UCB$S_TT_MULTI(R5),R4 : IS THIS CR PAST END OF PROMPT?
0254 774 BGEQU 112$ : YES THEN FORCE LINEFEED
0256 775 : In the case where we are sending a clear to end of line sequence (signaled
0256 776 : by SKIPCRLF being set) then
0256 777 : we don't want the linefeed to show up if there is one after a leading
0256 778 : <CR>. The case where there is a leading linefeed is handled by
0256 779 : setting SKIPLF in the EDITREAD routine.
0256 780 :
```

51 78 A5 DO
54 30 A1 3C
2C A1 2C B144 9E
30 A1 B4
32 A1 B4
3C 0122 C5 07 E1
41 11

51 78 A5 DO
54 34 A1 3C
54 4A A144 9E
54 00D8 C5 D1
22 1E

54	78	A5	0000004B	8F	C1	0256	781	IF NOT STATE SKIPCRLF,115\$:	IF NO LINEFEEDS WANTED THEN
						025A	782	CLR STATE <SKIPCRLF>	:	CLEAR NO LINEFEED
						025E	783	ADDL3 #TTY\$LB_DATA+1,UCB\$LB_SVAPTE(R5),R4	:	GET THE ADDRESS OF THE
						0267	784	CMPL UCB\$LB_TT_MULTI(R5),R4	:	FIRST CHARACTER AND SEE IF THIS IS IT
						026C	785	BNEQU 115\$:	NO THEN HANDLE NORMALLY
						026E	786	SET STATE <SKIPLF>	:	OTHERWISE SKIP THE LINEFEED
						0272	787	BRB 115\$:	AND HANDLE NORMALLY.
						0274	788			
						0274	789	110\$:		
						0274	790	IF STATE <SKIPCRLF>,115\$		
						0278	791	SET STATE <SENDLF,SKIPLF>	:	SET STATE TO FORCE LF AND SKIP NEXT REAL O
						027F	792	115\$:		
						027F	793	CLRW UCB\$WB_TT_CURSOR(R5)	:	RESET HORIZON
						0283	794	CMPL UCB\$B_TT_LASTC(R5), -	:	OPTIMIZATION: WAS LAST CHAR
						0288	795	#TTY\$C_CR	:	A CARRIAGE RETURN?
						0288	796	BEQL DROP	:	YES, DON'T OUTPUT ANOTHER
						028A	797	MOVB UCB\$B_TT_CRFILL(R5),UCB\$B_TT_FILL(R5)	:	SET UP FILL
						0291	798	BEQL 120\$:	IF EQL THEN OUTPUT
						0293	799	SET STATE FILL	:	OTHERWISE OUTPUT
						0296	800	BRW OUTPUTANDWAIT		
						0299	801	DROP:		
								BRW GETNEXTCHAR	:	CONTINUE IN NEXT LOGICAL STATE

54 78 A5 0000004B 8F C1 0256 781 IF NOT STATE SKIPCRLF,115\$: IF NO LINEFEEDS WANTED THEN
54 00D8 C5 D1 025A 782 CLR STATE <SKIPCRLF> : CLEAR NO LINEFEED
11 12 025E 783 ADDL3 #TTY\$LB_DATA+1,UCB\$LB_SVAPTE(R5),R4 : GET THE ADDRESS OF THE
0B 11 0267 784 CMPL UCB\$LB_TT_MULTI(R5),R4 : FIRST CHARACTER AND SEE IF THIS IS IT
11 12 026C 785 BNEQU 115\$: NO THEN HANDLE NORMALLY
0B 11 026E 786 SET STATE <SKIPLF> : OTHERWISE SKIP THE LINEFEED
11 12 0272 787 BRB 115\$: AND HANDLE NORMALLY.
0B 11 0274 788
0B 11 0274 789 110\$:
0B 11 0274 790 IF STATE <SKIPCRLF>,115\$
0B 11 0278 791 SET STATE <SENDLF,SKIPLF> : SET STATE TO FORCE LF AND SKIP NEXT REAL O
0B 11 027F 792 115\$:
0B 11 027F 793 CLRW UCB\$WB_TT_CURSOR(R5) : RESET HORIZON
0B 11 0283 794 CMPL UCB\$B_TT_LASTC(R5), - : OPTIMIZATION: WAS LAST CHAR
0B 11 0288 795 #TTY\$C_CR : A CARRIAGE RETURN?
0B 11 0288 796 BEQL DROP : YES, DON'T OUTPUT ANOTHER
0B 11 028A 797 MOVB UCB\$B_TT_CRFILL(R5),UCB\$B_TT_FILL(R5) : SET UP FILL
0B 11 0291 798 BEQL 120\$: IF EQL THEN OUTPUT
0B 11 0293 799 SET STATE FILL : OTHERWISE OUTPUT
0B 11 0296 800 BRW OUTPUTANDWAIT
0B 11 0299 801 DROP:
0B 11 0299 801 BRW GETNEXTCHAR : CONTINUE IN NEXT LOGICAL STATE


```
029C 803 .SBTTL CTRLZ - output control-Z
029C 804
029C 805 :++
029C 806 : CTRLZ - OUTPUT A CONTROL Z BASED ON THE OPERATION
029C 807 :
029C 808 : FUNCTIONAL DESCRIPTION:
029C 809 :
029C 810 : IF THE CURRENT OPERATION IS A READ AND THE ^Z IS A TERMINATOR THE ECHO ^Z.
029C 811 :
029C 812 : INPUTS:
029C 813 :
029C 814 : R2 = ADDRESS OF THE UNIT STATE VECTOR
029C 815 : R3 = CONTROL Z
029C 816 : R5 = UCB ADDRESS
029C 817 :
029C 818 : OUTPUTS:
029C 819 :
029C 820 : R2 = ADDRESS OF THE UNIT STATE VECTOR
029C 821 : R3 = CONTROL Z
029C 822 : R4 = ADDRESS OF THE MULTIECHO STRING FOR CONTROL Z IF APPROP.
029C 823 : R5 = UCB ADDRESS
029C 824 :--
029C 825 CTRLZ:
029C 826 IF_STATE WRITE,300$ : CONTROL Z
02A0 827 : IF THERE IS A WRITE ACTIVE THEN
02A0 828 : ECHO NORMALY
02A4 829 IF_NOT_STATE READ,300$ : OUTPUT 26(8) IF NOT READ
02A8 830 IF_NOT_STATE EOL,300$ : ^Z IF TERMINATOR
02AF 831 MOVL TTY$A,EXITCHO,R4 : ADDRESS MULTIECHO STRING
02B4 832 BBC #TT2$V,DECCRT,UCB$$_DEVDEPN2(R5),299$ :
02B8 833 SET STATE NINTMULTI : DON'T ALLOW THIS TO BE INTERUPTED
02BC 834 299$: MOVL 4(R4),R4 : OUTPUT THE DECCRT CONTROL-Z
02BF 835 BRW STRMULTI : GET THE ACTUAL STRING ADDRESS
02C2 836 300$: BRW OUTPUTANDWAIT : START OUTPUT
```

54 00000000'EF DO 02A8 830
08 48 A5 1D E1 02AF 831
54 04 A4 DE 02B8 833
54 64 DO 02BC 834
0D9E 31 02BF 835
FE83 31 02C2 836

```
02C5 838 .SBTTL ESCAPE - format an escape character
02C5 839
02C5 840 :++
02C5 841 : ESCAPE - FORMAT A ESCAPE BASED ON OPERATION AND TARGET UNIT
02C5 842 :
02C5 843 : FUNCTIONAL DESCRIPTION:
02C5 844 :
02C5 845 : THIS ROUTINE FORMATS ESCAPES.
02C5 846 :
02C5 847 : READ OPERATION:
02C5 848 :
02C5 849 : A '$' IS ECHOED IF THE ESCAPE IS A TERMINATOR.
02C5 850 :
02C5 851 : WRITE OPERATION:
02C5 852 :
02C5 853 : THE ESCAPE IS OUTPUT. ON TERMINALS WITH THE CHARACTERISTIC TTSM ESCAPE,
02C5 854 : THE REMAINDER OF THE SEQUENCE IS CHECKED FOR SYNTACTIC CORRECTNESS.
02C5 855 :
02C5 856 : INPUTS:
02C5 857 :
02C5 858 : R2 = ADDRESS OF THE UNIT STATE VECTOR
02C5 859 : R3 = ESCAPE
02C5 860 : R5 = UCB ADDRESS
02C5 861 :
02C5 862 : OUTPUTS:
02C5 863 :
02C5 864 : R2 = ADDRESS OF THE UNIT STATE VECTOR
02C5 865 : R3 = ESCAPE OR '$'
02C5 866 : R5 = UCB ADDRESS
02C5 867 :
02C5 868 : --
02C5 869 : ESCAPE:
02C5 870 : IF NOT STATE EOL,250$ : OUTPUT ESCAPE STRING
02C9 871 : IF STATE <MULTI,WRITE,EDITREAD>,250$; MULTIECHO THEN ASSUME SEQUENCE : ALSO IF NOT TERMINATOR
02D0 872 : MOVZBL #TTY$C_DOLLAR,R3 : OTHERWISE ECHO DOLLAR SIGN
02D3 873 : BSBW ADJUST_CURSOR : ADJUST CURSOR
02D6 874 : BRW OUTPUTANDWAIT : AND OUTPUT
02D9 875 :
02D9 876 : : OUTPUT ESCAPE ON WRITE OR CONTROL R
02D9 877 :
02D9 878 : 250$:
02D9 879 : BBS #TT2$V_ANSICRT,UCB$L_DEVDEPND2(R5),255$; BR IF ANSI TERMINAL
02DE 880 : IF STATE <ESCAPE>,255$ : IS THIS ESCAPE MODE
02E2 881 : CMPB #TT$VT5X,UCB$B_DEVTYPE(R5);VT5X OR VT100 TERMINAL? :
02E7 882 : BLEQU 252$ : If LEQU, then maybe.
02E9 883 : BRB 260$ : Otherwise, no.
02EB 884 :
02EB 885 : 252$:
02EB 886 : CMPB #TT$VT100+32,UCB$B_DEVTYPE(R5); :
02F0 887 : BGTRU 255$ : If GTRU, then yes.
02F2 888 : BRB 260$ : Otherwise, no.
02F4 889 :
02F4 890 : 255$:
02F4 891 : SET STATE <ESC_0> : OUTPUTTING AN ESCAPE SEQUENCE
02F9 892 : PUSH R1 : SAVE R1
02FB 893 : BSBW ESCINIT : INIT THE ESCAPE SEQUENCE RULES
02FE 894 : MOV R1,UCB$B_TT_ESC_0(R5) :
0303 894 : POPL R1 : RESTORE R1
```

53 24 9A 02D0 871
FED1 30 02D3 872
FE6F 31 02D6 873

16 48 A5 18 E0 02D9 878
41 A5 40 8F 91 02DE 879
02 1B 02E2 880
23 11 02E7 881
02EB 882
02EB 883
41 A5 80 8F 91 02EB 884
02 1A 02F0 885
1A 11 02F2 886
02F4 887
02F4 888
02F4 889
02F4 890
51 DD 02F9 891
019E 30 02FB 892
0104 C5 51 90 02FE 893
51 8ED0 0303 894

		0306	895			
		0306	896	IF NOT STATE <MULTI>,260\$:	MULTIECHOING AND EDITING
		030A	897	IF STATE <EDITING>,270\$:	DON'T ZERO CURSOR
		030E	898	260\$:		
00FC C5	B4	030E	899	CLRW UCBSW TT CURSOR(R5)	:	ZERO CURSOR (COLUMN) POSITION
FE33	31	0312	900	BRW OUTPUTANDWAIT	:	FALL INTO OUTPUT AND WAIT


```
0315 902 .SBTTL LINEFEED - format a line feed
0315 903
0315 904 :++
0315 905 : LINEFEED - FORMAT LINE FEED FOR TARGET UNIT
0315 906 :
0315 907 : FUNCTIONAL DESCRIPTION:
0315 908 :
0315 909 : THIS ROUTINE SETS UP THE PROPER FILL FOR A LINE FEED ON THE TARGET
0315 910 : UNIT AND ADJUSTS THE CURSOR AND VERTICAL LINE COUNT.
0315 911 :
0315 912 : INPUTS:
0315 913 :
0315 914 : R2 = ADDRESS OF THE UNIT STATE VECTOR
0315 915 : R3 = TTY$C_LF
0315 916 : R5 = UCB ADDRESS
0315 917 :
0315 918 : OUTPUTS:
0315 919 :
0315 920 : R2 = ADDRESS OF THE UNIT STATE VECTOR
0315 921 : R3 = TTY$C_LF
0315 922 : R5 = UCB ADDRESS
0315 923 :--
0315 924 : .ENABLE LSB
0315 925 : LINEFEED:
138 04 A2 0D E4 0315 926 : BBSC #TTY$V_ST_SKIPLF,4(R2),227$; SKIP AND SET OFF SKIP CONDITION
0315 927 :
0315 928 : NOTE THAT BECAUSE OF ACBB'S SIGNED BRANCH IT IS NOT APPROPRIATE HERE.
0315 929 :
0315 930 : LFOUTPUT:
0315 931 : TSTW UCB$W_TT_CURSOR(R5) : LINE FEED AT CURSOR 0 IS NEWLINE
0315 932 : BNEQ 210$ : IF NEQ THEN NO NL POSSIBLE
0315 933 : SET STATE NL : SET NEW LINE
0315 934 210$: INCB UCB$B_TT_LINE(R5) : ADJUST VERTICAL COUNT
0315 935 : CMPB UCB$B_TT_LINE(R5),UCB$L_DEVDEPEND+3(R5);
0315 936 : BLSSU 215$ : IF LSSU THEN NO PAGE CROSS
0315 937 : CLRB UCB$B_TT_LINE(R5) : RESET VERTICAL POSITION
0315 938 215$: IF NOT STATE_WRITE,220$ : BR IF NOT WRITE
0315 939 : MOVL UCB$L_TT_WRTBUF(R5),R4 : Address current write buffer.
0315 940 : MOVL TTY$L_WB_IRP(R4),R4 : Get associated IRP address.
0315 941 : BEQL 220$ : Branch if no IRP.
0315 942 : INCW IRP$L_MEDIA(R4) : COUNT LINE
0315 943 220$: MOVB UCB$B_TT_LFFILL(R5),UCB$B_TT_FILL(R5); INSERT TO CAUSE FILL
0315 944 : BEQL 225$ : OUTPUT IF NO FILL NEEDED
0315 945 : SET STATE FILL : OUTPUT FILL CHARACTERS
0315 946 225$: BRW OUTPUTANDWAIT
0315 947 227$: BRW DROP
0315 948 : .DISABLE LSB
```

00FC C5 B5 031A 931
04 12 031E 932
00FE C5 96 0324 933
47 A5 00FE C5 91 0328 934
04 1F 032E 935
00FE C5 94 0330 936
00D4 C5 D0 0334 937
54 24 A4 D0 0338 938
03 13 033D 939
38 A4 B6 0341 940
0102 C5 00F7 C5 90 0343 941
03 13 0346 942
FDF3 31 034F 943
FF41 31 0352 944
0355 945
0358 946

```
0358 950 .SBTTL TAB - Output a tab.
0358 951
0358 952 ++
0358 953 TAB - OUTPUT A TAB ON THE TARGET TERMINAL BASED ON CURSOR POSITION
0358 954
0358 955 FUNCTIONAL DESCRIPTION:
0358 956
0358 957 THIS ROUTINE IS ENTERED TO OUTPUT A TAB ON THE TARGET UNIT.
0358 958 IF THE TERMINAL HAS MECHANICAL TAB THEN THE TAB GOES DIRECT.
0358 959 OTHERWISE, THE CURSOR POSITION IS USED TO CALC. HOW MANY BLANKS
0358 960 TO OUTPUT TO MOVE THE CURSOR TO THE NEXT TAB STOP.
0358 961
0358 962 INPUTS:
0358 963
0358 964 R2 = ADDRESS OF THE UNIT STATE VECTOR
0358 965 R3 = TTY$C TAB
0358 966 R5 = UCB ADDRESS
0358 967
0358 968 OUTPUTS:
0358 969
0358 970 R2 = ADDRESS OF THE UNIT STATE VECTOR
0358 971 R3 = TTY$C TAB
0358 972 R4 = ADDRESS OF TAB MULTIECHO STRING IF APPROP.
0358 973 R5 = UCB ADDRESS
0358 974
0358 975 TAB: EXTZV #0,#3,UCB$W_TT_CURSOR(R5),R4: GET HORIZON POINTER
035F 976 IF STATE TABEXPAND,525$ : EXPAND THE TABS
0363 977 BBS #TT$V_MECHTAB,UCB$B_DEVDEPEND(R5),530$: OUTPUT IF MECHANICAL HELP
0368 978 525$: SUBW3 R4,#8,R1 : SETUP THE LENGTH OF THE TAB
036C 979 MOVAB W+TTY$A_TAB+1,R' : ADDRESS STRING TO OUTPUT
0371 980 BRW STRMULTI_1 : START MULTIPLE OUTPUT
0374 981
0374 982 : MECHANICAL TAB TERMINAL
0374 983
0374 984 : THIS ROUTINE ASSUMES THAT THE TABS ARE SET ON 8'S
0374 985
0374 986 530$: CMPB #TT$VT5X,UCB$B_DEVTYPE(R5): IN RANGE FOR SPECIAL CASE VT5X
0379 987 BGTRU 535$ : IF GTR THEN NO
037B 988 CMPB #TT$VT100+32,UCB$B_DEVTYPE(R5):
0380 989 BLEQU 535$
0382 990 SUBW3 UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5),R3: DISTANCE FROM RIGHT
0389 991 BLEQU 534$ : IF LEQU THEN SKIP TEST
038B 992 CMPW #8,R3 : IN LAST TAB SPACE?
038E 993 BGEQU 525$ : IF TRUE THEN SKIP MECH TAB
0390 994 534$: MOVZBL #TTY$C_TAB,R3
0393 995 535$: BICW R4,UCB$W_TT_CURSOR(R5) : SETUP CURSOR STOP ON 8
0393 996 ADDW #8,UCB$W_TT_CURSOR(R5)
039D 997 CLR_STATE - : No longer at newline; haven't
039D 998 <NL_WRAP> : just wrapped.
03A2 999 BRW OUTPUTANDWAIT
03A5 1000
03A5 1001 : FOR LOCAL TABS, TRACK POSITION, BUT DON'T ECHO
03A5 1002
03A5 1003
03A5 1004 TAB_LOCAL:
03A5 1005 EXTZV #0,#3,UCB$W_TT_CURSOR(R5),R4: GET HORIZON POINTER
03AC 1006 BICW R4,UCB$W_TT_CURSOR(R5) : SETUP CURSOR STOP ON 8

54 00FC C5 03 00 EF 0358 975 TAB: EXTZV #0,#3,UCB$W_TT_CURSOR(R5),R4: GET HORIZON POINTER
OC 44 A5 08 EO 035F 976 IF STATE TABEXPAND,525$ : EXPAND THE TABS
51 08 54 A3 0363 977 BBS #TT$V_MECHTAB,UCB$B_DEVDEPEND(R5),530$: OUTPUT IF MECHANICAL HELP
54 0001'CF 9E 0368 978 525$: SUBW3 R4,#8,R1 : SETUP THE LENGTH OF THE TAB
OCFA 31 036C 979 MOVAB W+TTY$A_TAB+1,R' : ADDRESS STRING TO OUTPUT
0371 980 BRW STRMULTI_1 : START MULTIPLE OUTPUT
0374 981
0374 982 : MECHANICAL TAB TERMINAL
0374 983
0374 984 : THIS ROUTINE ASSUMES THAT THE TABS ARE SET ON 8'S
0374 985
0374 986 530$: CMPB #TT$VT5X,UCB$B_DEVTYPE(R5): IN RANGE FOR SPECIAL CASE VT5X
0379 987 BGTRU 535$ : IF GTR THEN NO
037B 988 CMPB #TT$VT100+32,UCB$B_DEVTYPE(R5):
0380 989 BLEQU 535$
0382 990 SUBW3 UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5),R3: DISTANCE FROM RIGHT
0389 991 BLEQU 534$ : IF LEQU THEN SKIP TEST
038B 992 CMPW #8,R3 : IN LAST TAB SPACE?
038E 993 BGEQU 525$ : IF TRUE THEN SKIP MECH TAB
0390 994 534$: MOVZBL #TTY$C_TAB,R3
0393 995 535$: BICW R4,UCB$W_TT_CURSOR(R5) : SETUP CURSOR STOP ON 8
0393 996 ADDW #8,UCB$W_TT_CURSOR(R5)
039D 997 CLR_STATE - : No longer at newline; haven't
039D 998 <NL_WRAP> : just wrapped.
03A2 999 BRW OUTPUTANDWAIT
03A5 1000
03A5 1001 : FOR LOCAL TABS, TRACK POSITION, BUT DON'T ECHO
03A5 1002
03A5 1003
03A5 1004 TAB_LOCAL:
03A5 1005 EXTZV #0,#3,UCB$W_TT_CURSOR(R5),R4: GET HORIZON POINTER
03AC 1006 BICW R4,UCB$W_TT_CURSOR(R5) : SETUP CURSOR STOP ON 8

53 42 A5 00FC C5 A3 0382 990 SUBW3 UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5),R3: DISTANCE FROM RIGHT
53 05 1B 0389 991 BLEQU 534$ : IF LEQU THEN SKIP TEST
53 08 B1 038B 992 CMPW #8,R3 : IN LAST TAB SPACE?
53 D8 1E 038E 993 BGEQU 525$ : IF TRUE THEN SKIP MECH TAB
00FC C5 54 AA 0393 995 535$: BICW R4,UCB$W_TT_CURSOR(R5) : SETUP CURSOR STOP ON 8
00FC C5 08 A0 0393 996 ADDW #8,UCB$W_TT_CURSOR(R5)
FDA3 31 039D 997 CLR_STATE - : No longer at newline; haven't
03A2 998 <NL_WRAP> : just wrapped.
03A5 1000
03A5 1001 : FOR LOCAL TABS, TRACK POSITION, BUT DON'T ECHO
03A5 1002
03A5 1003
03A5 1004 TAB_LOCAL:
03A5 1005 EXTZV #0,#3,UCB$W_TT_CURSOR(R5),R4: GET HORIZON POINTER
03AC 1006 BICW R4,UCB$W_TT_CURSOR(R5) : SETUP CURSOR STOP ON 8
```

TTYCHARO
V04-000

M 6
- Terminal driver character output routi 16-SEP-1984 02:21:31 VAX/VMS Macro V04-00 Page 26
TAB - Output a tab. 5-SEP-1984 04:16:19 [TTDRVR.SRC]TTYCHARO.MAR;1 (16)

00FC C5 08 A0 03B1 1007
03B6 1008
03B6 1009
FC47 31 03B8 1010

ADDW #8,UCBSW_TT_CURSOR(R5) ;
CLR_STATE -
<NL,WRAP>
BRW GETNEXTCHAR ; GET NEXT OUTPUT CHARACTER


```
03BE 1012 .SBTTL VTAB, FORM - output a vertical tab, or form feed
03BE 1013
03BE 1014 :++
03BE 1015 : VTAB - OUTPUT A VERTICAL TAB
03BE 1016 : FORM - OUTPUT A FORM FEED
03BE 1017
03BE 1018 FUNCTIONAL DESCRIPTION:
03BE 1019
03BE 1020 THIS ROUTINE SETS UP THE PROPER OUTPUT STRING FOR A VTAB OF FORM FEED
03BE 1021 ON THE TARGET UNIT. VTAB TRANSLATES TO 4 LINE FEEDS. AND FORM FEED
03BE 1022 TO MULTIPLE LINE FEEDS BASED ON THE PRESENCE OF MECHANICAL HELP
03BE 1023 AND THE CURRENT VERTICAL LINE POSITION. TO AVOID THE NECESSITY
03BE 1024 FOR A LARGE NUMBER OF LF DATA BYTES THE FORM FEED CAUSES A MULTI
03BE 1025 ECHO STRING OF 4 LINE FEEDS FOLLOWED BY ANOTHER FORM FEED TO BE
03BE 1026 OUTPUT UNTIL A PAGE CROSS TAKES PLACE.
03BE 1027
03BE 1028 INPUTS:
03BE 1029
03BE 1030 R2 = ADDRESS OF THE UNIT STATE VECTOR
03BE 1031 R3 = C_VTAB OR C_FF
03BE 1032 R5 = UCB ADDRESS
03BE 1033
03BE 1034 OUTPUTS:
03BE 1035
03BE 1036 R2 = ADDRESS OF THE UNIT STATE VECTOR
03BE 1037 R5 = UCB ADDRESS
03BE 1038 --
54 0000'CF 9E 03BE 1039 VTAB: MOVAB W^TTY$A_VTAB,R4 ; SET UP OUTPUT
0C9A 31 03C3 1040 BRW STRTMULTI ; START MULTIPLE OUTPUT
03C6 1041
03C6 1042 : FORM FEED
03C6 1043
07 44 A5 13 E1 03C6 1044 FORM: BBC #TT$V_MECHFORM, - ; BRANCH IF TERMINAL DOES NOT
03CB 1045 UCB$B_DEVDEPEND(R5), - ; SUPPORT MECHANICAL FORM FEEDS
03CB 1046 505$
00FE C5 94 03CB 1047 CLR B UCB$B_TT_LINE(R5) ; RESET LINE POSITION
FD76 31 03CF 1048 BRW OUTPUTANDWAIT
54 D4 03D2 1049 505$: CLRL R4 ; SEND MULTIPLE LINE FEEDS
54 47 A5 00FE C5 83 03D4 1050 SUB B3 UCB$B_TT_LINE(R5),UCB$B_DEVDEPEND+3(R5),R4 ; GET NUMBER TO END OF PAG
04 54 91 03DB 1051 CMP B R4,#4 ; OUTPUT THE LF'S IN GROUPS OF 8
0C 1A 03DE 1052 BGTRU 520$ ; IF GTRU THEN MORE THAN 4 TO OUTPUT
51 54 01 A1 03E0 1053 510$: ADD W3 #1,R4,R1 ; GET THE NUMBER OF LINEFEEDS
54 0001'CF 9E 03E4 1054 MOVAB W^TTY$A_FORM+1,R4 ; GET THE ADDRESS OF THE STRING
0C82 31 03E9 1055 BRW STRTMULTI 1 ; START OUTPUT
54 0000'CF 9E 03EC 1056 520$: MOVAB W^TTY$A_LONGFORM,R4 ; GET ADDRESS TO STRING
0C6C 31 03F1 1057 BRW STRTMULTI
```

TTYCHARO
V04-000

- Terminal driver character output rout^{J 6} 16-SEP-1984 02:21:31 VAX/VMS Macro V04-00 Page 28
DISPATCHER SERVICE ROUTINES 5-SEP-1984 04:16:19 [TTDRVR.SRC]TTYCHARO.MAR;1 (18)
03F4 1059 .SBTTL DISPATCHER SERVICE ROUTINES

```

03F4 1061 .SBTTL BACKSPACING - OUTPUT N BACKSPACES
03F4 1062 :++
03F4 1063 : BACKSPACING:
03F4 1064 :
03F4 1065 : DESCRIPTION:
03F4 1066 : SEND A GIVEN NUMBER OF BACKSPACES TO THE TERMINAL
03F4 1067 :
03F4 1068 : INPUTS:
03F4 1069 : R2 = UNIT STATE VECTOR ADDRESS
03F4 1070 : R3 = UCB ADDRESS
03F4 1071 : UCBSW_TT_MULTILEN = NUMBER OF BACKSPACES TO OUTPUT
03F4 1072 :--
03F4 1073 : BACKSPACING:
0100 C5 B7 03F4 1074 : DECW UCBSW_TT_BSLEN(R3) ; SUBTRACT ONE AND OUTPUT THE BACKSPACE
      06 19 03F8 1075 : BLSS 10$ ; NO MORE THEN FINISH UP
53 08 9A 03FA 1076 : MOVZBL #TTYSC_BS,R3 ; SETUP BACKSPACE
      FD87 31 03FD 1077 : BRW FORMAT_CHAR ; AND FORMAT THE CHARACTER
      FBFF 31 0400 1078 10$: CLR_STATE BACKSPACE ; CLEAR THE STATE WHEN DONE
      0403 1079 : BRW GETNEXTCHAR ; AND GET THE NEXT CHARACTER
      0406 1080
  
```

```
0406 1082 .SBTTL CURSOROVRFLOW - insert newline to handle end of line
0406 1083 :++
0406 1084 :CURSOROVRFLOW - INSERT A NEWLINE IN THE OUTPUT STREAM
0406 1085 :
0406 1086 :FUNCTIONAL DESCRIPTION:
0406 1087 :
0406 1088 :THIS ROUTINE IS ENTERED TO INJECT A CR/LF IN THE OUTPUT
0406 1089 :STREAM REGARDLESS OF THE CURRENT STATE.
0406 1090 :
0406 1091 :INPUTS:
0406 1092 :
0406 1093 :R2 = ADDRESS OF THE UNIT STATE VECTOR
0406 1094 :R5 = UCB ADDRESS
0406 1095 :
0406 1096 :OUTPUTS:
0406 1097 :
0406 1098 :R2 = ADDRESS OF THE UNIT STATE VECTOR
0406 1099 :R3 = TTY$C CR
0406 1100 :R4 = READ PACKET ADDRESS
0406 1101 :R5 = UCB ADDRESS
0406 1102 :--
0406 1103 :CURSOROVRFLOW:
0406 1104 :CLR_STATE CURSOR : CLEAR THE CONDITION
0406 1105 :SET_STATE WRAP : Signal WRAPPING.
0406 1106 :IF NOT STATE READ,99$ : IF WE ARE NOT READING THEN HANDLE NORMALY
0412 1107 :IF STATE RDVERIFY,98$ : NO CURSOR ORERFLOW ON READ VERIFY'S
0416 1108 :MOVL UCB$S_SVAPTE(R5),R4 : OTHERWISE GET THE PACKED ADDRESS
041A 1109 :IF NOT STATE CTRLR,97$ : IF WE ARE CONTROL-RING HANDLE NORMALY
2C A4 00DB C5 D1 041E 1110 :CMPL UCB$S_TT_MULTI(R5),TTY$S_RB_LIN(R4); IF WE AREN'T ON THE LAST LINE
: 05 1B 0424 1111 :BLEQU 99$ : THEN DON'T WORRIE ABOUT LINREST
: 32 A4 B5 0426 1112 97$: :TSTW TTY$W_RB_LINREST(R4) : ARE WE IN THE MIDDLE OF A LINE
: 0D 12 0429 1113 :BNEQ 98$ : YES THEN LEAVE THE STATUS QUOE
53 0D 9A 042B 1114 99$: :MOVZBL #TTY$C_CR,R3 : SET FIRST CHARACTER IS CARRIAGE RETURN
: 042E 1115 :SET_STATE <SENDLF,SKIPLF> : FORCE A FREE LINEFEED
: FDD7 31 0435 1116 :BRW CROUTPUT : CONTINUE WITH CARRIAGE RETURN CODE
: FESE 31 0438 1117 98$: :BRW DROP
```



```
043B 1119 .SBTTL EOLSEEN - handle end of line condition
043B 1120
043B 1121 :++
043B 1122 : EOLSEEN - END OF LINE SEEN
043B 1123 :
043B 1124 : FUNCTIONAL DESCRIPTION:
043B 1125 :
043B 1126 : THIS ROUTINE IS ENTERED AFTER AN END OF LINE CHARACTER HAS
043B 1127 : TERMINATED A READ RECORD AND THE ECHO OF THAT CHARACTER IS COMPLETE.
043B 1128 :
043B 1129 : THE ACTION IS TO COMPLETE THE READ OPERATION AND IF NECESSARY,
043B 1130 : SEND AN XOFF.
043B 1131 :
043B 1132 : INPUTS:
043B 1133 :
043B 1134 : R2 = ADDRESS OF THE UNIT STATE VECTOR
043B 1135 : R5 = UCB ADDRESS
043B 1136 :
043B 1137 : OUTPUTS:
043B 1138 :
043B 1139 : R2 = ADDRESS OF THE UNIT STATE VECTOR
043B 1140 : R5 = UCB ADDRESS
043B 1141 : --
043B 1142 : EOLSEEN: ; END READ OPERATION
043B 1143 :
043B 1144 : IN ORDER FOR THE FINAL CHARACTERS OF A READ VERIFY READ TO ECHO
043B 1145 : PROPERLY EDIT READ MUST BE ALLOWED TO COMPLETE, THIS CAN BE DONE
043B 1146 : IN TWO WAYS, FIRST CHANGING THE STATE DISPATCHER TO ALLOW EDIT READ
043B 1147 : TO HAVE HIGHER PRIORITY THEN EOL. THIS FIXES EOL IN THE STATE TABLE
043B 1148 : FOR NOW AND FOREVER. BY PUTTING THIS CODE HERE IT ALLOWS EOL TO
043B 1149 : MOVE TO ANY PLACE IN THE STATE DISPATCHER.
043B 1150 :
043B 1151 : IF NOT STATE RDVERIFY,5$ ; LET EDIT READ COMPLETE UNDER
043B 1152 : IF NOT STATE EDITREAD,5$ ; READ VERIFY
043B 1153 : BRW EDITREAD ;...
043B 1154 :
0905 31 0443 1153
0446 1154
7C A5 01 9B 0446 1155 5$: MOVZBW #SS$ NORMAL,UCB$W_BOFF(R5); SET STATUS
FBB3' 30 044A 1156 BSBW TTY$READONE ; COMPLETE THE I/O
FBB0 31 044D 1157 BRW TTY$GETNEXTCHAR ; CONTINUE
```

```
0450 1159 .SBTTL FILLING - continue outputting fill characters
0450 1160
0450 1161 :++
0450 1162 : FILLING - CONTINUE FILL OUTPUT
0450 1163 :
0450 1164 : FUNCTIONAL DESCRIPTION:
0450 1165 :
0450 1166 : THIS ROUTINE IS ENTERED WHILE THE TRMSV FILL STATE IS ON.
0450 1167 : THE FILL COUNT IS DECREMENTED AND IF NON 0 A FILL IS OUTPUT
0450 1168 : IF THE COUNT GOES TO 0 THEN THE FILL STATE IS CLEARED.
0450 1169 :
0450 1170 : INPUTS:
0450 1171 :
0450 1172 : R2 = ADDRESS OF THE UNIT STATE VECTOR
0450 1173 : R5 = UCB ADDRESS
0450 1174 :
0450 1175 : OUTPUTS:
0450 1176 :
0450 1177 : R2 = ADDRESS OF THE UNIT STATE VECTOR
0450 1178 : R3 = 0
0450 1179 : R5 = UCB ADDRESS
0450 1180 : --
0450 1181 : FILLING:
0102 C5 97 0450 1182 DECB UCB$B_TT_FILL(R5) ; OUTPUT CONTINUING FILLS
03 12 0454 1183 BNEQ 25$ ; ADJUST FILL COUNT
0456 1184 CLR STATE FILL ; IF NEQ THEN NOT DONE YET
53 D4 0459 1185 25$: CLRC R3 ; SET CONDITION OFF
FCEF 31 045B 1186 BRW OUTPUTANDWAIT1 ; SET FILL CHARACTER
; OUTPUT THE CHARACTER
```

```
045E 1188 .SBTTL MULTIECHOING - Continue outputting multiecho sequence
045E 1189
045E 1190 :++
045E 1191 : MULTIECHOING - CONTINUE MULTIECHO STRING OUTPUT
045E 1192 :
045E 1193 : FUNCTIONAL DESCRIPTION:
045E 1194 :
045E 1195 : THIS ROUTINE IS ENTERED WHEN TTY$V ST MULTI IS SET. THE NEXT
045E 1196 : CHARACTER IN THE STRING ADDRESSED BY UCB$L TT_MULTI IS OUTPUT.
045E 1197 : IF THE NEXT CHARACTER IS ZERO THEN THE STRING OUTPUT IS COMPLETE
045E 1198 : AND THE MULTIECHO CONDITION IS RESET FOLLOWED BY A RETURN TO
045E 1199 : THE OUTPUT STATE ANALYSIS ROUTINE IN GETNEXTCHAR.
045E 1200 : IF A LENGTH IS SPECIFIED IN UCB$W TT_MULTILEN, THEN IT IS USED TO DETECT
045E 1201 : THE END OF STRING.
045E 1202 :
045E 1203 : INPUTS:
045E 1204 :
045E 1205 : R2 = ADDRESS OF THE UNIT STATE VECTOR
045E 1206 : R5 = UCB ADDRESS
045E 1207 :
045E 1208 : UCB$W TT_MULTILEN = ALTERNATE LENGTH
045E 1209 :
045E 1210 : OUTPUTS:
045E 1211 :
045E 1212 : R2 = ADDRESS OF THE UNIT STATE VECTOR
045E 1213 : R3 = CHARACTER TO OUTPUT ( FALL THROUGH TO ECHOFORMAT )
045E 1214 : R5 = UCB ADDRESS
045E 1215 : --
045E 1216 MULTIECHOING:
045E 1217 DECW UCB$W TT_MULTILEN(R5) ; CONTINUE STRING MULTI ECHO
0462 1218 BGEQ 20$ ; DECREMENT THE COUNT
0464 1219 MOVL UCB$L TT_SMLT(R5),UCB$L TT_MULTI(R5); RESTORE SAVED ADDRESS
0468 1220 MOVW UCB$W TT_SMLTLEN(R5),UCB$W TT_MULTILEN(R5);AND LENGTH
0472 1221 BGTR 10$
0474 1222 CLR_STATE <MULTI,NINTMULTI> ; NO MORE THEN RESET THE STATE
047C 1223 BRW GETNEXTCHAR ; AND GET THE NEXT CHARACTER
047F 1224 10$: CLRW UCB$W TT_SMLTLEN(R5) ; CLEAR THE SAVED LENGTH
0483 1225 DECW UCB$W TT_MULTILEN(R5) ; SUBTRACT OUT THIS CHARACTER
0487 1226 20$: MOVZBL @UCB$C TT_MULTI(R5),R3 ; GET NEXT MULTI ECHO CHARACTER
048C 1227 INCL UCB$L TT_MULTI(R5) ; ADJUST POINTER
0490 1228 BRW FORMAT_CHAR ;
```

00DC C5 B7 045E 1217
23 18 0462 1218
00E0 C5 D0 0464 1219
00DE C5 B0 0468 1220
0B 14 0472 1221
FB86 31 0474 1222
00DE C5 B4 047C 1223
00DC C5 B7 047F 1224
53 00D8 D5 9A 0483 1225
00D8 C5 D6 0487 1226
FCF4 31 048C 1227
0490 1228

```
0493 1230 .SBTTL SENDLINEFEED - output a line feed
0493 1231
0493 1232 :++
0493 1233 : SENDLINEFEED - INSERT LINE FEED IN OUTPUT STREAM
0493 1234 :
0493 1235 : FUNCTIONAL DESCRIPTION:
0493 1236 :
0493 1237 : THIS ROUTINE IS USED TO INSERT A LINE FEED IN THE OUTPUT STREAM.
0493 1238 :
0493 1239 : INPUTS:
0493 1240 :
0493 1241 : R2 = ADDRESS OF THE UNIT STATE VECTOR
0493 1242 : R5 = UCB ADDRESS
0493 1243 :
0493 1244 : OUTPUTS:
0493 1245 :
0493 1246 : R2 = ADDRESS OF THE UNIT STATE VECTOR
0493 1247 : R3 = TTY$C_LF
0493 1248 : R5 = UCB ADDRESS
0493 1249 :--
0493 1250 SENDLINEFEED:
0493 1251 CLR STATE SENDLF
0493 1252 MOVZBL #TTY$C_LF,R3
0493 1253 BRW LFOUTPUT
: SEND OUT A LINE FEED
: CLEAR CONDITION
: SET TO OUTPUT LINE FEED
: CONTINUE IN LINE FEED CODE
```

53 0A 9A
FE7E 31


```
049C 1255 .SBTTL ESCAPE SEQUENCE PARSING SERVICES
049C 1256 :++
049C 1257 :
049C 1258 : ESCINIT - INIT ESCAPE SEQUENCE RULES
049C 1259 :
049C 1260 : INPUTS:
049C 1261 : R3 = CHARACTER THAT IS INTRODUCER
049C 1262 :
049C 1263 : OUTPUTS:
049C 1264 : R1 = RULE TO START AT (BYTE)
049C 1265 :
049C 1266 : ESCINIT::
049C 1267 : CLRL R1 ; USE THE NORMAL RULES
049E 1268 : CMPB R3,#^X80 ; IF NOT AN 8 BIT CHARACTER
04A2 1269 : BLSSU 10$
04A4 1270 : MOVB TTY$A_ESCINIT-^X80[R3],R1; ELSE MOVE THRU THE TABLE
04AC 1271 10$: RSB
```

51 80 8F 51 D4 049C 1267 CLRL R1 ; USE THE NORMAL RULES
53 08 53 91 049E 1268 CMPB R3,#^X80 ; IF NOT AN 8 BIT CHARACTER
08 1F 04A2 1269 BLSSU 10\$
51 FFFFFFF80'EF43 90 04A4 1270 MOVB TTY\$A_ESCINIT-^X80[R3],R1; ELSE MOVE THRU THE TABLE
05 04AC 1271 10\$: RSB

```
04AD 1273 .SBTTL ESCSYNTAX -- CHECK ESCAPE SEQUENCE SYNTAX
04AD 1274
04AD 1275
04AD 1276 :++
04AD 1277 :FUNCTIONAL DESCRIPTION:
04AD 1278 :   This routine checks the syntax of escape sequences, one
04AD 1279 :   character at a time.
04AD 1280 :
04AD 1281 :INPUTS:
04AD 1282 :   R3 contains the next character to be checked for correct syntax
04AD 1283 :   UCB$B_TT_ESC(R5) contains a offset to the next syntax
04AD 1284 :   rule in the TTY$A_ESCAPE syntax table for input data.
04AD 1285 :   UCB$B_TT_ESC_O(R5) contains the offset for output data.
04AD 1286 :   Assume R4 is available
04AD 1287 :
04AD 1288 :OUTPUTS:
04AD 1289 :   Condition codes are set:
04AD 1290 :   CC = POSITIVE means the character is syntactically correct
04AD 1291 :   CC = EQUAL means syntactic correctness and sequence is complete
04AD 1292 :   CC = NEGATIVE means error in parsing sequence
04AD 1293 :
04AD 1294 :   If CC = POSITIVE the UCB$B_TT_ESC (ESC_O) offset is updated.
04AD 1295 :
04AD 1296 :--
04AD 1297 :
04AD 1298 :INPUT ESCAPE SEQUENCES
04AD 1299 :
04AD 1300 ESCSYNTAX:
04AD 1301     PUSHL R4 ; SAVE R4
04AF 1302     MOVZBL UCB$B_TT_ESC(R5),R4 ; GET CURRENT STATE
04B4 1303     BSBB E_SYNTAX
04B6 1304     MOV B4,UCB$B_TT_ESC(R5) ; SAVE STATE
04BB 1305     POPL R4 ; AND RESTORE R4
04BE 1306     TSTB UCB$B_TT_ESC(R5) ; SET THE CONDITION CODES
04C2 1307     RSB
04C3 1308
04C3 1309 :
04C3 1310 :OUTPUT ESCAPE SEQUENCES
04C3 1311 :
04C3 1312 ESCSYNTAX_O:
04C3 1313     MOVZBL UCB$B_TT_ESC_O(R5),R4 ; GET CURRENT STATE
04C8 1314     BSBB E_SYNTAX
04CA 1315     MOV B4,UCB$B_TT_ESC_O(R5) ; SAVE CURRENT STATE
04CF 1316     RSB
04D0 1317
04D0 1318 :
04D0 1319 :
04D0 1320 :GENERAL SEQUENCE HANDLER
04D0 1321 :
04D0 1322 :
04D0 1323 E_SYNTAX::
04D0 1324
0000'CF44 53 91 04D0 1325 10$: CMPB R3, W^TTY$A_ESCAPE[R4] ; check range of character
0001'CF44 0F 1F 04D6 1326     BLSSU 20$ ; branch if not this rule
0001'CF44 53 91 04D8 1327     CMPB R3, W^TTY$A_ESCAPE+1[R4] ; lower than high limit?
0001'CF44 07 1A 04DE 1328     BGTRU 20$ ; look to next rule
04E0 1329
```

```
54 0002'CF44 9A 04E0 1330      MOVZBL W'ATTY$A_ESCAPE+2[R4],- ; character is valid
                                R4                                     ; save offset to next rule
                                05 04E6 1331
                                04E6 1332
                                04E7 1333      RSB                     ; offset=POS => next rule exists
                                04E7 1334      ; offset=0 => end of sequence
                                04E7 1335
                                04E7 1336
                                04E7 1337 : Continue looking for correct sequence rule
                                04E7 1338
0002'CF44 95 04E7 1339 20$: TSTB W'ATTY$A_ESCAPE+2[R4] ; any next rule?
                                05 13 04EC 1340      BEQL 30$ ; no; bad syntax
54 03 C0 04EE 1341      ADDL #3, R4 ; offset to next rule
                                DD 11 04F1 1342      BRB 10$ ; continue syntax check
                                04F3 1343 30$:
54 01 CE 04F3 1344      MNEGL #1, R4 ; error in sequence
                                05 04F6 1345      RSB
```

```
04F7 1347 .SBTTL MOVEREADATA - MOVE CHARACTER FROM TYPEAHEAD TO READ BUFFER
04F7 1348 :++
04F7 1349 : MOVEREADATA -- MOVE CHARACTER FROM TYPEAHEAD BUFFER TO READ BUFFER
04F7 1350 :
04F7 1351 : FUNCTIONAL DESCRIPTION:
04F7 1352 :
04F7 1353 : THIS ROUTINE MOVES A CHARACTER FROM THE TYPEAHEAD BUFFER AND STARTS
04F7 1354 : THE ECHO.
04F7 1355 :
04F7 1356 : NON-IMMEDIATE ACTION CONTROL SEQUENCES ARE HANDLED HERE.
04F7 1357 :
04F7 1358 : BEFORE RETURNING A CHARACTER FOR ECHO IT IS CONVERTED TO ITS
04F7 1359 : MULTIPLE ECHO STRING IF APPROPRIATE. IN THIS CASE THE CHARACTER
04F7 1360 : RETURNED IS THE FIRST OF THE MULTIPLE ECHO CHARACTERS.
04F7 1361 :
04F7 1362 : INPUTS:
04F7 1363 : R5 = UCB ADDRESS
04F7 1364 :
04F7 1365 : OUTPUTS:
04F7 1366 :
04F7 1367 : R3 = CHARACTER IF ANY (CC = EQL )
04F7 1368 : R5 = UCB ADDRESS
04F7 1369 : --
04F7 1370 :
04F7 1371 : GOTO TABLE
04F7 1372 :
51 011F 31 04F7 1373 TAB_NOTA: BRW NOTA
58 A5 D0 04FA 1374 TAB_EOLSEEN: MOVL UCBSL_IRP(R5),R1
38 A1 B4 04FE 1375 CLRW IRPSL_MEDIA(R1)
FF37 31 0501 1376 BRW EOLSEEN
0153 31 0504 1377 TAB_DISMISS: BRW DISMISS
0101 31 0507 1378 TAB_BUFEMPTY: BRW BUFEMPTY
022E 31 050A 1379 TAB_INDELST: BRW INDELST
0C1B 31 050D 1380 TAB_PASSALL: BRW PASSALL
FAF2 31 0510 1381 TAB_GETNEXT: BRW GETNEXTCHAR
02D8 31 0513 1382 TAB_QUOTE: BRW QUOTE
02EA 31 0516 1383 TAB_EDITINGCHAR: BRW EDITINGCHAR
53 95 0519 1384 TAB_ESCINPROG: TSTB R3 ; NO NULLS IN ESCAPE SEQUENCES
E7 13 051B 1385 BEQL TAB_DISMISS
013F 31 051D 1386 BRW ESCINPROG
012A 31 0520 1387 TAB_CVTLOW: BRW CVTLOW
022C 31 0523 1388 TAB_OPEN: BRW OPEN
0526 1389 :
0526 1390 : MAIN LINE MOVE OUT OF BUFFER CODE
0526 1391 MOVEREADATA:
0526 1392 :
54 78 A5 D0 0526 1393 .ENABLE LSB
14 A4 D4 052A 1394 MOVL UCBSL_SVAPTE(R5),R4 ; SETUP R4 TO CONTAIN THE ADDRESS OF THE
40 A4 3C A4 B1 052D 1395 CLRL TTYSL_RB_ECHSTR(R4) ; CLEAN OUT ANY UNUSED ECHO STRINGS
C6 1E 052D 1395 CMPW TTYSW_RB_TXTOFF(R4),TTYSW_RB_TXTSIZ(R4); TEST BUFFER FULL
0532 1396 BGEQU TAB_EOLSEEN ; BUFFER FULL THEN END
0534 1397 :
0534 1398 : MOVE CHARACTER OUT OF TYPEAHEAD BUFFER
0534 1399 :
54 00E4 C5 D0 0534 1400 MOVL UCBSL_TT_TYPAHD(R5),R4 ; ADDRESS TYPEAHEAD BUFFER
BC 13 0539 1401 BEQL TAB_NOTA ; NO BUFFER THEN THERE IS A FORK
053B 1402 ; PROCESS WAITING SO LET IT GO.
OC A4 B7 053B 1403 DECW TTYSW_TA_INAHD(R4) ; take the character out of typeahead
```



```
53 04 C7 19 053E 1404      BLSS    TAB_BUFEMPTY      : IF EQL THEN NO
04 B4 9A 0540 1405      MOVZBL @TTYSL_TA_GET(R4),R3 : GET THE CHARACTER
0544 1406      IF_STATE DEL,TAB_INDELST : ARE WE IN DELETE STATE
0548 1407      :
0548 1408      : ACTUALLY REMOVE CHARACTER FROM BUFFER
0548 1409      :
0548 1410      :
0548 1411      : CHECK FOR BUFFER WRAP AROUND
0548 1412      :
06 04 A4 10 A4 F2 0548 1413 40$: AOBLS TAB_END(R4),TTYSL_TA_GET(R4),60$: POINTER PAST END?
04 A4 0118 C4 9E 054E 1414      MOVAB    TTYSL_TA_DATA(R4),TTYSL_TA_GET(R4); RESET POINTER
0554 1415      :
0554 1416      : SKIP TESTS FOR SECOND CONTROL SEQUENCE SET IF PASSALL
0554 1417      :
0554 1418      :
54 78 A5 D0 0554 1419 60$: MOVL    UCBSL_SVAPTE(R5),R4 : SETUP R4 TO CONTAIN THE ADDRESS OF THE
0558 1420      IF_STATE <PASSALL,NOFLTR>,TAB_PASSALL; IN PASSALL THEN PASS ALL
055F 1421      IF_STATE QUOTING,TAB_QUOTE : ARE WE QUOTING
51 00000000'EF43 9A 0563 1422 67$: MOVZBL TTYSA_CCLIST[R3],R1 : GET THE TOKEN
A9 12 056B 1423      BNEQ    TAB_EDITINGCHAR : SPECIAL ACTION THEN HANDLE SPECIALLY
056D 1424      :
056D 1425      : INSERT THE CHARACTER
056D 1426      :
056D 1427 151$: IF STATE <ESC>,TAB_ESCINPROG; IF IN AN ESCAPE SEQUENCE THEN HANDLE AS SUCH
0571 1428 INSERT_CHAR:
0571 1429 152$: MOVL    UCBSL_IRP(R5),R1 : GET THE IRP ADDRESS
A6 20 A1 08 E0 0575 1430      BBS     #IOSV_CVTLOW,IRPSW_FUNC(R1),TAB_CVTLOW; BR IF CONVERT LOWER TO UPPER
3D 1C B4 53 E0 057A 1431 155$: BBS     R3,@TTYSL_RB_TERM(R4),TERMFOUND; IS THIS CHARACTER A TERMINATOR
51 30 A4 3C 057F 1432      MOVZWL TTYSW_RB_LINOFF(R4),R1 : GET THE OFFSET INTO THE BUFFER
32 A4 B5 0583 1433      TSTW    TTYSW_RB_LINREST(R4) : DO WE HAVE TO OPEN THE BUFFER
9B 12 0586 1434      BNEQ    TAB_OPEN : NO THEN HANDLE NORMALLY
2C B441 53 90 0588 1435 159$: MOVB    R3,@TTYSL_RB_LIN(R4)[R1]; INSERT CHARACTER
3C A4 B6 058D 1436      INCW    TTYSW_RB_TXTOFF(R4) : ADJUST POINTER
30 A4 B6 0590 1437      INCW    TTYSW_RB_LINOFF(R4) : ADD TO COUNT
0593 1438      :
0593 1439      : CHECK FOR FULL BUFFER
0593 1440      :
0593 1441 CHECK_BUFFER:
40 A4 3C A4 B1 0593 1442 210$: CMPW    TTYSW_RB_TXTOFF(R4),TTYSW_RB_TXTSIZ(R4); TEST BUFFER FULL
13 1E 0598 1443      BGEQU    212$ : BUFFER FULL THEN END
059A 1444      :
059A 1445      : ECHO THE CHARACTER IF NECESSARY
059A 1446      :
059A 1447 230$: IF STATE <NOECHO,ESC,EDITREAD>,250$: IF NOECHO OR ESC THEN BRANCH
FBDD 31 05A7 1448      BRQ     FORMAT_CHAR :
FA58 31 05AA 1449 240$: BRW     GETNEXTCHAR : VECTOR TO TRANSFER MORE CHARACTERS
05AD 1450      :
05AD 1451      : BUFFER OVERFLOWED THEN TERMINATE
05AD 1452      :
05AD 1453 212$: MOVL    UCBSL_IRP(R5),R1 : GET THE IRP ADDRESS
51 58 A5 D0 05B1 1454      CLRW    IRPSL_MEDIA(R1) : SET NO TERMINATOR
20 A1 1000 8F AA 05B4 1455      BICW    #IOSM_TRMNOECHO,IRPSW_FUNC(R1); FORCE POSSIBLE ECHO
18 11 05BA 1456      BRB     221$ : AND CONTINUE
05BC 1457      :
05BC 1458      : TERMINATOR FOUND BUT BUFFER NOT FULL
05BC 1459      :
05BC 1460 TERMFOUND:
```

```
51 58 A5 D0 05BC 1461      MOVL   UCBSL_IRP(R5),R1      : GET THE IRP ADDRESS
38 A1 53 B0 05C0 1462      MOVW   R3,IRPSL_MEDIA(R1)      : SAVE LAST CHARACTER FOR STATUS
      3A A1 B6 05C4 1463      INCW   IRPSL_MEDIA+2(R1)      : SET TERMINATOR SIZE
51 5C A4 3C 05C7 1464      MOVZWL  TTYSW_RB_TXTOFF(R4),R1      : GET THE OFFSET TO THE LAST CHARACTER
D0 B441 53 90 05CB 1465      MOVW   R3,@TTYSC_RB_TXT(R4)[R1] : PUT THE LAST CHARACTER AWAY
51 58 A5 D0 05D0 1466      MOVL   UCBSL_IRPTR5T,R1      : GET THE IRP ADDRESS BACK
CD 20 A1 0C E0 05D4 1467 221$: SET_STATE <EOC>      : SET END OF LINE SEEN
      FFBA 31 05D8 1468 225$: BBS     #IOSV_TRMNOECHO,IRPSW_FUNC(R1),240$; BR IF NOT TERM NOECHO
      05DD 1469      BRW     230$
      05E0 1470      : TEST FOR ECHO NEEDS
      05E0 1471      :
      05E0 1472      :
      05E0 1473 250$: IF STATE <EDITREAD,ESC>,240$      : CONTINUE IF ESCAPE
      05ED 1474      BBC     #TT2$V_LOCALECHO,-
      05EF 1475      UCBSL_DEVDEPND2(R5),240$      : NO FORMATTING IF NOT LOCAL ECHO
51 58 A5 D0 05F2 1476      MOVL   UCBSL_IRP(R5),R1      : GET THE IRP ADDRESS
AF 20 A1 06 E0 05F6 1477      BBS     #IOSV_NOECHO,-
      05FB 1478      IRPSW_FUNC(R1),240$      : OR READ NOECHO
      FBE9 31 05FB 1479      BRW     FORMAT_LOCAL      : OTHERWISE, LOCAL ECHO TERMINAL, FORMAT
```

```
                                .SBTTL MOVEREADATA SERVICE ROUTINES
05FE 1481
05FE 1482 :
05FE 1483 : TYPEAHEAD BUFFER EXHAUSTED
05FE 1484 :
05FE 1485 25$: IF NOT STATE NOECHO,21$ : NO ECHO READ NECESSARY TO OPTIMIZE
0602 1486 SET_STATE PRE : SET THE BYPASS TYPEAHEAD BUFFER STATE
DD 11 0606 1487 BRB 21$ : CONTINUE IN THE NORMAL CODE PATH
0027 31 0608 1488 :
0608 1489 20$: BRW XON : CONTINUE
OC A4 B4 060B 1490 BUFEMPTY:
060B 1491 CLRW TTYSW TA INAHD(R4) : FIX THE COUNT TO REFLECT 0
060E 1492 IF STATE <PASALL,NOFLTR>,25$ :
0615 1493 21$: IF STATE <TYPFUL>,20$ : BR IF TYPEAHEAD FULL
0619 1494 NOTA:
0619 1495 :
0619 1496 : THE TYPEAHEAD BUFFER HAS NOW BEEN EMPTIED INTO THE SYSTEM BUFFER.
0619 1497 : IF PROCESSING A READ WITH ZERO SECOND TIMEOUT, RETURN THE DATA
0619 1498 : TO THE USER IMMEDIATELY.
0619 1499 :
00B0 C5 3C 68 A5 01 E1 0619 1500 BBC #UCBSV TT TIMO,UCBSW DEVSTS(R5),DISMISS; BR IF NOT READ WITH TIMEOUT
00000000'EF D1 061E 1501 CMPL TTYSW MAXTIME,UCBSL TT_RDUE(R5); ZERO SECOND TIMEOUT?
31 12 0627 1502 BNEQ DISMISS : BR IF NOT.
7C A5 022C 8F B0 0629 1503 MOVW #SS$ TIMEOUT,UCBSW_BOFF(R5); SET TIMEOUT COMPLETION STATUS.
F9CE' 30 062F 1504 BSBW TTYSREADONE : COMPLETE REQUEST.
0632 1505
```

```

0632 1507 :++
0632 1508 : XON - SEND XON
0632 1509 :
0632 1510 : FUNCTIONAL DESCRIPTION:
0632 1511 :
0632 1512 : THIS ROUTINE IS USED TO SEND AN XON
0632 1513 :
0632 1514 : INPUTS:
0632 1515 :
0632 1516 : R2 = ADDRESS OF THE UNIT STATE VECTOR
0632 1517 : R5 = UCB ADDRESS
0632 1518 :
0632 1519 : OUTPUTS:
0632 1520 :
0632 1521 : R2 = ADDRESS OF THE UNIT STATE VECTOR
0632 1522 : R3 = XON
0632 1523 : R5 = UCB ADDRESS
0632 1524 : --
0632 1525 : XON:
0632 1526 : CLR STATE <TYPFUL> ; SEND AN XON
0636 1527 : IF_NOT_STATE <OVRFLO>,10$ ; SKIP IF NO OVERFLOW ERROR
063A 1528 :
063A 1529 : CLR STATE <OVRFLO> ; RESET CONDITION
7C AS 0838 BF B0 063E 1530 : MOVW #SS$ DATAOVERUN,JCB$W_BOFF(R5); COMPLETE READ IN ERROR
F9B9' 30 0644 1531 : BSBW TTY$READONE
F9B6' 30 0647 1532 :
F9B8 31 0647 1533 10$: BSBW TTY$XON
064A 1534 : BRW GETNEXTCHAR

```


TTYCHARO
V04-000

- Terminal driver character output rout^{L 7} 16-SEP-1984 02:21:31 VAX/VMS Macro V04-00 Page 43
MOVEREADATA SERVICE ROUTINES 5-SEP-1984 04:16:19 [TTDRVR.SRC]TTYCHARO.MAR;1 (31)

```
064D 1536  
064D 1537 :  
064D 1538 : CONVERT CHARACTER TO UPPER CASE  
064D 1539 :  
03 0000'CF43 03 E1 064D 1540 CVTLOW: BBC #TTYSV.CH_LOWER,W^TTYS^_TYPE[R3],154$; BR IF NOT LOWER CHARACTER  
53 20 AA 0654 1541 BICW #^X020,R3  
FF20 31 0657 1542 154$: BRW 155$; CONVERT TO UPPER CASE  
155$
```

TTYCHARO
V04-000

- Terminal driver character output rout^{M 7}i 16-SEP-1984 02:21:31 VAX/VMS Macro V04-00
MOVEREADATA SERVICE ROUTINES 5-SEP-1984 04:16:19 [TTDRVR.SRC]TTYCHARO.MAR:1 Page 44
(32)

```
065A 1544 :  
065A 1545 : DISMISS INTERRUPT  
065A 1546 :  
065A 1547 DISMISS:  
010B C5 94 065A 1548 CLR B UCB$B_TT_OUTYPE(R5) : SET NO RETURN CHARACTER.  
05 065E 1549 RSB : and dismiss the interrupt
```

```
065F 1551 :  
065F 1552 : ESCAPE SEQUENCE IN PROGRESS - CHECK SYNTAX  
065F 1553 :  
065F 1554 ESCINPROG:  
51 3C A4 95 065F 1555 TSTB R3 : NO NULLS IN ESCAPE SEQUENCES  
00 B441 3C A4 13 0661 1556 BEQL 961$ : SO DROP IT  
51 3C A4 90 0663 1557 MOVZWL TTY$W_RB TXTOFF(R4),R1 : GET THE PLACE TO STORE THE ESCAPE  
3C A4 B6 0667 1558 MOVB R3,TTY$C_RB TXT(R4),R1 : AND PLACE THE CHARACTER IN THE RIGHT PLACE  
51 58 A5 D0 066C 1559 INCW TTY$W_RB TXTOFF(R4) : AND INCREMENT THE WORLD  
3A A1 B6 066F 1560 MOVL UCBSL_IRP(R5),R1 : GET THE IRP ADDRESS  
FE34 30 0673 1561 INCW IRP$M_MEDIA+2(R1) : ADJUST SIZE OF ESCAPE SEQUENCE  
2D 14 0676 1562 BSBW ESCSYNTAX : CHECK CHAR AGAINST SYNTAX  
04 13 0679 1563 BGTR 960$ : VALID CHAR, CONTINUE INPUT  
067B 1564 BEQL 197$ : VALID CHAR, SEQUENCE COMPLETE  
067D 1565 :  
067D 1566 : ESCAPE SYNTAX ERROR  
067D 1567 :  
067D 1568 195$: SET_STATE BADESC : SET BAD ESCAPE  
0681 1569 197$: CLR_STATE ESC : SET SEQUENCE DONE  
3C A4 3A A1 A2 0686 1570 SUBW IRP$M_MEDIA+2(R1),TTY$W_RB TXTOFF(R4) : ADJUST TRANSFER SIZE  
068B 1571 IF_STATE <NOFILTER,BADESC>,199$ : BAD SEQUENCE THEN HANDLE NORMALLY OR  
0693 1572 : NOFILTER READ  
53 7E 8F 91 0693 1573 CMPB #A/~/,R3 : IS THIS A FUNCTION KEY  
4C 12 0697 1574 BNEQ 910$ : NO THEN HANDLE APROPRIATELY  
48 A4 15 B1 0699 1575 CMPW #21,TTY$W_RB_ESCTKN(R4) : IS THIS THE EXIT KEY  
13 12 069D 1576 BNEQ 920$ : NO THEN TRY NORMAL KEY  
53 1A 9A 069F 1577 MOVZBL #TTY$C_CTRLZ,R3 : SETUP A CONTROL-Z  
38 A1 D4 06A2 1578 CLRL IRP$M_MEDIA(R1) : CLEAN THE ESCAPE SEQUENCE UP  
018B 31 06A5 1579 BRW 162$ : AND GO OFF TO DISPATCH  
0078 31 06A8 1580 960$: BRW 198$  
06AB 1581 :  
06AB 1582 :  
06AB 1583 199$:  
06AB 1584 SET_STATE EOL : SET UP END OF LINE  
F953 31 06AF 1585 961$: BRW GETNEXTCHAR : THEN GET THE NEXT CHARACTER  
06B2 1586 :  
06B2 1587 :  
06B2 1588 : FUNCTION KEY DETECTED CHANGE FUNCTION KEY TO TOKEN  
06B2 1589 : BUT ONLY IF WE ARE EDITING  
06B2 1590 :  
06B2 1591 920$: IF NOT_STATE <EDITING>,199$ : NO EDITING THE GO AND RETURN THE SEQUENCE  
0000'8F 48 A4 B1 06B6 1592 CMPW TTY$W_RB_ESCTKN(R4),#TTY$K_MAXESCTKN : IS IT OUT OF RANGE  
57 1A 06BC 1593 BGTRU 190$ : YES THEN RETURN IT  
51 48 A4 3C 06BE 1594 MOVZWL TTY$W_RB_ESCTKN(R4),R1 : GET THE OFFSET  
51 00000000'EF41 9A 06C2 1595 MOVZBL TTY$A_FCNTKN[R1],R1 : AND MAKE THAT INTO A TOKEN  
06CA 1596 930$:  
51 95 06CA 1597 TSTB R1 : IS THIS VALID  
47 13 06CC 1598 BEQL 190$ : NO THEN RETURN THE SEQUENCE  
53 58 A5 D0 06CE 1599 191$: MOVL UCBSL_IRP(R5),R3 : GET THE IRP ADDRESS  
38 A3 D4 06D2 1600 CLRL IRP$M_MEDIA(R3) : CLEAR THE ESCAPE SEQUENCE OUT  
0D 51 91 06D5 1601 CMPB R1,#TTY$K_ET_UNUSED : IS THIS AN UNUSED ESCAPE SEQUENCE  
3F 13 06D8 1602 BEQL 970$ : YES THEN IGNORE IT...  
0B 51 91 06DA 1603 CMPB R1,#TTY$K_ET_RECALL : IS THIS A RECALL COMMAND  
03 12 06DD 1604 BNEQ 192$ : NO THEN CONTINUE NORMALLY  
53 02 9A 06DF 1605 MOVZBL #TTY$C_CTRLB,R3 : YES THEN MAKE IT A CONTROL-B AND CONTINUE  
0127 31 06E2 1606 192$: BRW 70$  
06E5 1607
```

```
03 3A A1 B1 06E5 1608 910$: IF NOT_STATE <EDITING>,199$ : AND WE ARE NOT EDITING DO THE SAME
                                06E9 1609 : CMPW IRP$-MEDIA+2(R1),#3 : IS THE SEQUENCE LONGER THAN 3 CHARACTERS
                                14 06ED 1610 : BGTR 190$ : YES THEN DO A MATCH ON IT
51 05 9A 06EF 1611 : MOVZBL #TTY$K_ET_BACK_CHAR,R1 : SETUP FOR BACK CHARACTER
53 44 8F 91 06F2 1612 : CMPB #^A/D/,R3 : IS IT A BACK CHARACTER
                                13 06F6 1613 : BEQL 930$ : YES THEN DISPATCH
51 06 9A 06F8 1614 : MOVZBL #TTY$K_ET_FORWARD_CHAR,R1 : GO FOWARD A CHARACTER
53 43 8F 91 06FB 1615 : CMPB #^A/C/,R3 : IS IT A FORWARD
                                13 06FF 1616 : BEQL 930$ : HANDLE IT
51 0B 9A 0701 1617 : MOVZBL #TTY$K_ET_RECALL,R1 : RECALL A COMMAND
53 41 8F 91 0704 1618 : CMPB #^A/A/,R3 : IS THIS RECALL?
                                C0 13 0708 1619 : BEQL 930$
06 20 A4 10 E0 070A 1620 : BBS #TRM$V_TM_NORECALL,TTY$-RB MOD(R4),190$: CHECK NORECALL
53 42 8F 91 070F 1621 : CMPB #^A/B/,R3 : IS THIS A DOWN ARROW?
                                B5 13 0713 1622 : BEQL 930$ : YES THEN DROP IT
                                0715 1623 :
                                0715 1624 : DON'T TERMINATE ON THIS SEQUENCE UNLESS HE REALY WANTS IT
                                0715 1625 : (THE MODE OF THE READ IS ESCAPE
                                0715 1626 :
0715 1627 190$: IF_STATE ESCAPE,199$ : IF THIS IS AN ESCAPE SEQUENCE READ THEN
0719 1628 970$: : TERMINATE ON ESCAPE SEQUENCES.
51 58 A5 D0 0719 1629 : MOVL UCB$-IRP(R5),R1 : MAKE SURE WE HAVE THE IRP ADDRESS
38 A1 D4 071D 1630 : CLRL IRP$-MEDIA(R1) : NO THEN CLEAN THE SEQUENCE OFF THE TERMINA
F8E2 31 0720 1631 : BRW GETNEXTCHAR : AND GET THE NEXT CHARACTER
                                0723 1632 :
                                0723 1633 : NORMAL ESCAPE SEQUENCE TERMINATION
                                0723 1634 :
                                0723 1635 :
53 30 91 0723 1636 198$: CMPB #^A/0/,R3 : IS THIS A NUMBER
                                14 0726 1637 : BGTR 900$ : NO THEN EXIT
53 39 91 0728 1638 : CMPB #^A/9/,R3 : ...
                                19 072B 1639 : BLSS 900$
48 A4 0A A4 072D 1640 : MULW #10,TTY$-RB_ESCTKN(R4) : SHIFT THE NUMBER OVER
53 30 A2 0731 1641 : SUBW #^A/0/,R3 : SUBTRACT OUT 0
48 A4 53 A0 0734 1642 : ADDW R3,TTY$-RB_ESCTKN(R4) : ADD IT IN
FE58 31 0738 1643 900$: BRW 210$
```



```
073B 1645 :  
073B 1646 : SEE IF DELETE SEQUENCE SHOULD BE TERMINATED  
073B 1647 :  
7F 8F 53 91 073B 1648 INDELST:CMPI R3,TTYSC_DELETE : IS THIS CHARACTER A DELETE?  
OE 13 073F 1649 BEQL 50$ : IF YES THEN GO ON  
OC A4 B6 0741 1650 INCW TTYSW_TA_INAHD(R4) : NO THEN PUT THE CHARACTER BACK  
0744 1651 CLR_STATE DEL : TERMINATE DELETE SEQUENCE  
0748 1652 :  
0748 1653 : TERMINATE DELETE SEQUENCE IF NOT IN NO-ECHO  
0748 1654 :  
53 5C 8F 9A 0748 1655 MOVZBL #^A/\/,R3 : SET OUTPUT CHARACTER  
FA38 31 074C 1656 BRW FORMAT_CHAR : FORMAT THE CHARACTER  
074F 1657 :  
FDF6 31 074F 1658 50$: BRW 40$  
0752 1659
```

```
0752 1661 :  
0752 1662 : OPEN THE BUFFER  
0752 1663 : OR JUST OVERSTRIKE  
0752 1664 :  
0752 1665 OPEN:  
0752 1666 :  
0752 1667 : SPREAD THE BUFFER TO PUT THIS CHARACTER IN PLACE  
0752 1668 :  
3F BB 0752 1669 : PUSHR #*M<R0,R1,R2,R3,R4,R5> : SAVE THE REGISTERS OVER THE MOVE  
51 D4 0754 1670 : CLRL R1 : SET FORWARD MODE  
07F9 30 0756 1671 : BSBW CHAR_SIZE : GET THE CHARACTER SIZE  
0759 1672 :  
0759 1673 : IF NOT AT THE END OF LINE THEN DON'T LET THE USER WRAP THE LINE  
0759 1674 :  
0759 1675 : IF STATE <WRAP,TABRIGHT>,194$  
38 A4 00FC C5 53 A1 0763 1676 169$: ADDW3 R3,UCB$W_TT CURSOR(R5),TTY$W_RB_CPZCUR(R4); AND ADD IT INTO THE CURR  
51 30 A4 3C 076A 1677 : MOVZWL TTY$W_RB_LINOFF(R4),R1 : GET THE LOCATION OF THE LAST CHARACTER  
14 A4 2C B441 9E 076E 1678 : MOVAB @TTY$C_RB_LIN(R4)[R1],TTY$C_RB_ECHSTR(R4); GET THE ADDRESS  
44 A4 01 B0 0774 1679 : MOVW #TTY$K_ER_CLRECHO,TTY$W_RB_MODE(R4); AND SETUP THE ECHO MODE  
0778 1680 : IF STATE NOECHO,193$ : DON'T ECHO IF NOT ECHOING...  
077C 1681 : SET STATE EDITREAD : AND SET THE EDIT READ STATE  
09 0C AE 91 0780 1682 193$: IF NOT STATE OVERSTRIKE,158$ : NOT IN OVERSTRIKE THEN HANDLE NORMALLY  
2D 13 0784 1683 : CMPB 12(SP),#TTY$C_TAB : WAS THIS CHARACTER A TAB?  
0788 1684 : BEQL 158$ : YES THEN INSERT IT.  
32 A4 B7 078A 1685 :  
3C A4 B7 078A 1686 : DECW TTY$W_RB_LINREST(R4) : IN OVERSTRIKE WE DELETE WHILE TYPING  
078D 1687 : DECW TTY$W_RB_TXTOFF(R4) : FAKE OUT THE REST OF THE LOGIC  
0790 1688 :  
01 53 D1 0790 1689 : CMPL R3,#1 : IS THIS A SHORT ECHOING CHARACTER  
09 14 B4 91 0793 1690 : BNEQ 168$ : NO THEN OUTPUT THE WHOLE THING  
0C 13 0795 1691 : CMPB @TTY$C_RB_ECHSTR(R4),#TTY$C_TAB; ARE WE ON TOP OF A TAB?  
44 A4 B4 0799 1692 : BEQL 157$ : YES THEN HANDLE SPECIALLY  
14 A4 D4 079B 1693 : CLRW TTY$W_RB_MODE(R4) : CLEAN OUT THE MODE FOR LATER  
079E 1694 : CLRL TTY$C_RB_ECHSTR(R4) : MAKE SURE THE STRING IS ZERO  
07A1 1695 : CLR_STATE EDITREAD : AND DON'T DO THE EDITREAD ECHO (ONLY  
07A5 1696 : : ONE CHARACTER NO NEED)  
53 38 A4 03 1A 11 07A5 1697 : BRB 168$ : AND DON'T BOTHER WITH THE MOVC  
00 EF 07A7 1698 157$: EXTZV #0,#3,TTY$W_RB_CPZCUR(R4),R3; GET THE LENGTH OF THE TAB  
53 D5 07AD 1699 : TSTL R3 : DELETE THE TAB WHEN IT EXPANDS TO ZERO  
1B 13 07AF 1700 : BEQL 171$ : MAKE IT DISAPPEAR  
32 A4 B6 07B1 1701 : INCW TTY$W_RB_LINREST(R4) : ELSE JUST INSERT THE CHARACTER  
3C A4 B6 07B4 1702 : INCW TTY$W_RB_TXTOFF(R4)  
51 D6 07B7 1703 158$: INCL R1 : INCREMENT THE REGISTER BY 1 TO GET THE  
07B9 1704 : : NEW POSITION  
2C B441 32 A4 28 07B9 1705 : MOVC3 TTY$W_RB_LINREST(R4),- : AND MOVE THE CHARACTERS OVER  
14 B4 07BC 1706 : @TTY$C_RB_ECHSTR(R4),@TTY$C_RB_LIN(R4)[R1]  
3F BA 07C1 1707 168$: POPR #*M<R0,R1,R2,R3,R4,R5> : RESTORE THE REGISTERS  
OB A4 01 32 A4 81 07C3 1708 : ADDB3 TTY$W_RB_LINREST(R4),#1,TTY$C_RB_ECHLEN(R4); AND LENGTH OF STRING TO  
FDBC 31 07C9 1709 : BRW 159$ : AND INSERT THE CHARACTER  
30 A4 B6 07CC 1710 171$: INCW TTY$W_RB_LINOFF(R4) : LOOK FROM THIS CHARACTER ON  
08BB 30 07CF 1711 : BSBW TABRIGHT : MAKE SURE THE TAB AFTER THIS POSITION  
30 A4 B7 07D2 1712 : : FLAG IS SET CORRECTLY.  
EA 11 07D2 1713 : DECW TTY$W_RB_LINOFF(R4) : KEEP LINOFF THE SAME AS WE STARTED WITH  
07D5 1714 : BRB 168$ : JOIN THE FLOW  
07D7 1715 :  
07D7 1716 : : ONCE WRAP IS DETERMINED THEN CHECK SPECIFICALLY IF THIS CHARACTER IS VALID  
07D7 1717 :
```

01	53	B1	07D7	1718	194\$:	IF NOT_STATE OVERSTRIKE,234\$:	DON'T EVER INSERT AFTER THE WRAP
	06	12	07DB	1719		CMPL	:	IS THIS A SINGLE CHARACTER EXPANSION
			07DE	1720		BNEQ	:	NO THEN DON'T INSERT IT
09	OC	AE	91	07E0	1721	CMPL	:	WAS THIS CHARACTER A TAB?
			05	12	07E4	1722	:	NO THEN OVERSTRIKE IT
			3F	BA	07E6	1723	:	RESTORE THE REGISTERS
			F81A	31	07E8	1724	:	AND GET THE NEXT CHARACTER
			FF75	31	07EB	1725	:	
					234\$:	POPR		
						BRW		
					235\$:	BRW		
						169\$		

```

07EE 1727 :
07EE 1728 : QUOTING CHARACTER DETECTED
07EE 1729 :
OD 04 A2 14 E4 07EE 1730 QUOTE: BBSC #TTY$V-ST EDITING,4(R2),65$; IF EDITING IS SET THEN CLEAR IT
07F3 1731 IF STATE ESC,65$ : IF ESCAPE THEN DON'T TURN OFF EDITING YET
07F7 1732 SET_STATE EDITING : IF IT NOT EDITING THEN SET EDITING
07FB 1733 CLR_STATE QUOTING : AND CLEAR QUOTING
FD60 31 0800 1734 65$: BRW 67$ : RETURN TO THE NORMAL PLACE

```



```
.SBTTL Specail input character dispatcher

0803 1736 :++
0803 1737 :
0803 1738 : SERVICE ROUTINES TO MOVE READATA
0803 1739 :
0803 1740 :
0803 1741 : SPECAIL EDITING CHARACTER HANDELING
0803 1742 :
0803 1743 EDITINGCHAR:
0803 1744 IF STATE <EDITING>,70$ : EDITING THEN NO RANGE CHECK NECESSARY
04 51 D1 0807 1745 CMPL R1,#TTY$K_EDITNORMAL : NOT EDITING THEN ANYTHING ADVANCED
27 14 080A 1746 BGTR 162$ : IS BAD
080C 1747 70$: CASE R1,TYPE=L,<-
080C 1748 151$,- : IS NORMAL CHARACTER (SHOULD NEVER BE USED)
080C 1749 CTRLU,- : IS CONTROL-U
080C 1750 CTRLR,- : INDICATES CONTROL-R
080C 1751 DELCHAR,- : INDICATES DELETE CHARCTER
080C 1752 ESCAPE CHAR,- : INDICATES ESCAPE CHARACTER
080C 1753 BACK CHAR,- : MOVE BACK ONE CHARACTER
080C 1754 FORWARD CHAR,- : MOVE FOWARD ONE CHARACTER
080C 1755 MOVE_EOL,- : MOVE TO THE END OF LINE
080C 1756 MOVE_BOL,- : MOVE TO THE BEGINNING OF LINE
080C 1757 DELETE_WORD,- : DELETE WORD TO THE LEFT
080C 1758 QUOTING,- : QUOTE CHARACTER
080C 1759 RECALL,- : RECALL THE LAST COMMAND
080C 1760 TOGGELINSOV- : TOGGEL INSERT/OVERSTRIKE MODE
080C 1761 >
082A 1762 IF_STATE TERMNORM,162$ : TERMINATE ON USUAL
082E 1763 : CHARACTERS
OE 51 D1 082E 1764 CMPL R1,#TTY$K_ET_TERMINATE : WAS THIS A REAL TERMINATOR CHARACTER
1E 12 0831 1765 BNEQ 165$ : NO THEN DON'T LET IT THRU
OF 1C B4 53 E0 0833 1766 162$: IF STATE <ESC>,166$ : IF IN AN ESCAPE SEQUENCE THEN LET IT THRU
OD 53 91 0837 1767 BBS R3,@TTY$LB_TERM(R4),164$ : IS THIS CHARACTER A TERMINATOR
53 91 083C 1768 CMPB R3,#TTY$C_CR : WE ARE PREPARED TO HANDLE <CR>
13 13 083F 1769 BEQL 800$ : SO LET IT THRU ANYWAY
0841 1770 IF STATE <QUOTING,EDITING>,165$ : NO THEN DROP IT IF WE ARE EDITING
FD22 31 084B 1771 167$: BRQ 151$
FD6E 31 084B 1772 164$: BRW TERMFOUND : TERMINATOR DETECTED
FE0E 31 084E 1773 166$: BRW ESCINPROG : ESCAPE SEQUENCE INPROGRESS
F7B1 31 0851 1774 165$: BRW GETNEXTCHAR : GET THE NEXT CHAR
32 A4 B5 0854 1775 800$: TSTW TTY$W_RB_LINREST(R4) : ARE WE AT THE END OF THE LINE
EF 13 0857 1776 BEQL 167$ : YES THEN CR IS LEGAL
F7A9 31 0859 1777 BRW GETNEXTCHAR
085C 1778
085C 1779
085C 1780
.disable lsb
```

```
085C 1782 :++
085C 1783 : RECALLING
085C 1784 :
085C 1785 : DESCRIPTION:
085C 1786 : THIS STATE IS USED WHEN THE LAST COMMAND TYPED IS REQUESTED.
085C 1787 : THIS ROUTINE WILL REMOVE CHARACTERS FROM THE RECALL BUFFER THEN
085C 1788 : ALLOW THE NORMAL INSERT CHARACTER LOGIC TAKE AFFECT. THE CHARACTERS
085C 1789 : IN THE RECALL BUFFER ARE PROCESSED AS IF THEY WERE NO FORMAT.
085C 1790 :
085C 1791 : INPUTS:
085C 1792 : R5 = UCB ADDRESS
085C 1793 :
085C 1794 : OUTPUTS:
085C 1795 : SEE MOVEREADATA
085C 1796 :--
085C 1797 : RECALLING:
51 00E4 C5 D0 085C 1798 : MOVL UCB$$_TT_TYPAHD(R5),R1 : GET THE ADDRESS OF THE TYPEAHEAD BUFFER
      25 13 0861 1799 : BEQL 20$ : NONE THEN LET MOVEREADATA HANDLE IT
54 78 A5 D0 0863 1800 : MOVL UCB$$_SVAPTE(R5),R4 : SETUP THE READ PACKET ADDRESS
53 0E A1 3C 0867 1801 : MOVZWL TTY$$_TA_RCLOFF(R1),R3 : GET THE ADDRESS OF THE NEXT CARACTER
53 18 A143 9A 086B 1802 : MOVZBL TTY$$_TA_RCL(R1)[R3],R3 : GET THE CHARACTER
      0E A1 B6 0870 1803 : INCW TTY$$_TA_RCLOFF(R1) : MOVE ON TO THE NEXT CHARACTER
14 A1 0E A1 B1 0873 1804 : CMPW TTY$$_TA_RCLOFF(R1),TTY$$_TA_RCLSIZ(R1); ARE WE FINISHED
      07 18 0878 1805 : BGEQ 5$ : YES THEN TURN OFF RECALL AND CONTINUE
40 A4 0E A1 B1 087A 1806 : CMPW TTY$$_TA_RCLOFF(R1),TTY$$_RB_TXTSIZ(R4); ARE WE GOING TO OVERFLOW
      04 19 087F 1807 : BLSS 10$ : THE USERS BUFFER
      FCE9 31 0881 1808 5$: CLR_STATE RECALL : YES THEN MAKE THIS THE LAST TIME
      FC9B 31 0885 1809 10$: BRW INSERT_CHAR : AND INSERT THE CHARACTER
      0888 1810 20$: BRW MOVEREADATA : ABORT EXIT
```

```
.SBTTL Post typeahead character action routines

088B 1812      :++
088B 1813      : BACK_CHAR
088B 1814      : Move backward a single character.
088B 1815      :--
088B 1816      : BACK_CHAR:
088B 1817      :
088B 1818      : TSTW      TTY$W_RB_CPZORG(R4)      : START AT ZERO?
088B 1819      : BEQL      10$      : YES THEN NO SPECAIL ACTION
088B 1820      : BSBW      ZERO CPZORG      : NO THEN MAKE SURE WE DO
088B 1821      : BSBL      10$: DECW      TTY$W_RB_LINOFF(R4)      : BACKUP A CHARACTER
088B 1822      : BLSS      50$      : AT THE END THEN DON'T ALLOW IT
088B 1823      : INCW      TTY$W_RB_LINREST(R4)      : UPDATE THE REST POINTER
088B 1824      : MOVZWL     TTY$W_RB_LINOFF(R4),R1      : GET THE OFFSET TO THE CHARACTER
088B 1825      : MOVZBL     @TTY$C_RB_LIN(R4)[R1],R3      : GET THE CHARACTER
088B 1826      : CMPB      #TTY$C_TAB,R3      : IS THIS A TAB
088B 1827      : BNEQ      20$      : NO THEN CONTINUE
088B 1828      : SET STATE TABRIGHT      : ELSE MARK THIS INFORMATION
088B 1829      : MNEGL     #1,R1      : GOING BACKWARDS TO
088B 1830      : BSBW      CHAR SIZE      : CHECK THE SIZE OF THE CHARACTER
088B 1831      : SUBW3     R3,U$BSW TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); UPDATE THE CURSOR POSITI
088B 1832      : IF STATE NOECHO,80$-
088B 1833      : BRQ       UPDATE_CURSOR      : SET THE CURSOR TO IT'S NEW LOCATION
088B 1834      :
088B 1835      :
088B 1836      : 50$: CLRW      TTY$W_RB_LINOFF(R4)      : RESET THE POINTER
088B 1837      : 60$: BRW       DROP      : AND DROP THE CHARACTER
```

```
08C8 1839 .enable lsb
08C8 1840
08C8 1841 :: CONTROL R PROCESSING
08C8 1842
08C8 1843 CTRLR: IF_STATE <ESC>,150$ : NO ACTION IF ESCAPE OR NO ECHO
08CC 1844 IF_NOT_STATE NOECHO,100$ : BRANCH IF ECHO
BBC 1845 #TT2$V_LOCALECHO,-
08D0 1846 UCB$$_DEVDEPND2(R5),150$; BRANCH IF NOT LOCAL ECHO
08D2 1847
08D5 1848 100$: MOVAB W^TTY$A_CTRLR,R4 : ADDRESS MULTIECHO STRING
54 0000'CF 9E 08D5 1849 110$: BBC #TT2$V_EDITING,-
OC E1 08DA 1849 UCB$$_DEVDEPND2(R5),130$ : NO THEN USE NORMAL ECHO STRING
17 44 A5 OC E1 08DF 1851 BBC #TT$V_SCOPE,UCB$$_DEVDEPEND(R5),130$; DON'T SKIP STUFF
08E4 1852 : ON A HARD COPY TERMINAL
51 78 A5 D0 08E4 1853 MOVL UCB$$_SVAPTE(R5),R1 : GET THE PACKET ADDRESS
3A A1 B5 08E8 1854 TSTW TTY$W_RB_CPZORG(R1) : WERE WE AT THE FIRST COLUMN WHEN
04 12 08EB 1855 BNEQ 120$ : WE STARTED THE READ
08ED 1856 SET_STATE <SKIPCRLF> : NO LINEFEEDS AFTER <CR>
08F1 1857 120$: BBC #TT2$V_ANSICRT,- : IS IT AN ANSI CRT
05 48 A5 E1 08F3 1858 UCB$$_DEVDEPND2(R5),130$; NO THEN USE NORMAL ECHO STRING
54 0000'CF 9E 08F6 1859 MOVAB W^TTY$A_ANSI_DEOL,R4 : YES THEN WE CAN USE CRT MODE
08FB 1860 130$:
08FB 1861 SET_STATE <EDITREAD> : SET CONTROL R STATE
075E 31 08FF 1862 BRW STRMULTI : START MULTIPLE OUTPUT SEQUENCE
0902 1863
FC21 31 0902 1864 150$: BRW MOVEREADATA
```



```
0905 1866 : CONTROL U PROCESSING
0905 1867 :
0905 1868 :
0239 30 0905 1869 10$: BSBW DELESCAPE : DELETE THE ESCAPE SEQUENCE
30 A4 B5 0908 1870 CTRLU: IF STATE ESC,10$ : IF IN ESCAPE SEQUENCE THEN DELETE ESCAPE
30 A4 13 090C 1871 TSTW TTY$W_RB_LINOFF(R4) : ANY DATA TO CONTROL-U?
69 13 090F 1872 BEQL 50$ : NO THEN DON'T DO THE WORK
3C A4 30 A4 A2 0911 1873 SUBW TTY$W_RB_LINOFF(R4),TTY$W_RB_TXTOFF(R4) : CLEAN OUT THE TEXT OFFSETO
08 A4 94 0916 1874 CLRB TTY$B_RB_ECHLEN(R4) : CLEAN THE EXTRA
38 A4 B4 0919 1875 CLRW TTY$W_RB_CPZCUR(R4) : MOVE TO THE BEGINNING OF THE LINE
3A A4 B5 091C 1876 TSTW TTY$W_RB_CPZORG(R4) : WAS THIS AT THE RIGHT
04 12 091F 1877 BNEQ 12$ : NO THEN USE THE OLD METHOD
44 A4 06 98 0921 1878 MOVZBW #TTY$K_ER_CLRREST,TTY$W_RB_MODE(R4) : EXIT WHEN NOT DOING ANYTHING EL
32 A4 B5 0925 1879 12$: TSTW TTY$W_RB_LINREST(R4) : IS THERE A REST OF THE LINE
26 13 0928 1880 BEQL 13$ : NO THEN HANDLE NORMALLY
3F B8 092A 1881 PUSHR #*M<R0,R1,R2,R3,R4,R5> : SAVE THE REGISTERS
51 30 A4 3C 092C 1882 MOVZWL TTY$W_RB_LINOFF(R4),R1 : GET THE NUMBER OF CHARACTERS
2C B4 2C B4 32 A4 28 0930 1883 MOVCL TTY$W_RB_LINREST(R4),@TTY$B_RB_LIN(R4) : AND RESTORE THE REGISTERS
3F B8 0938 1884 POPR #*M<R0,R1,R2,R3,R4,R5> : WAS THIS AT THE RIGHT
3A A4 B5 093A 1885 TSTW TTY$W_RB_CPZORG(R4) : NO THEN USE THE OLD METHOD
18 12 093D 1886 BNEQ 14$ : NO THEN USE THE OLD METHOD
44 A4 01 98 093F 1887 MOVZBW #TTY$K_ER_CLRECHO,TTY$W_RB_MODE(R4) : SETUP THE NEW ECHO MODE
14 A4 2C A4 D0 0943 1888 MOVL TTY$B_RB_LIN(R4),TTY$B_RB_ECHSTR(R4) : THE LOCATION OF THE FIRST CHAR
08 A4 32 A4 90 0948 1889 MOVW TTY$W_RB_LINREST(R4),TTY$B_RB_ECHLEN(R4) : AND THE LENGTH
38 A4 B4 094D 1890 CLRW TTY$W_RB_CPZCUR(R4) : AND GO ALL THE WAY TO THE BEGINNING
64 2C A4 D1 0950 1891 13$: CMPL TTY$B_RB_LIN(R4),TTY$B_RB_TXT(R4) : ARE WE ON THE FIRST LINE
04 12 0954 1892 BNEQ 14$ : NO THEN CONTINUE AS USUAL
44 A4 07 98 0956 1893 MOVZBW #TTY$K_ER_PRMECHO,TTY$W_RB_MODE(R4) : ECHO THE PROMPT TOO
30 A4 B4 095A 1894 14$: CLRW TTY$W_RB_LINOFF(R4) : NOW ZERO THE LINE OFFSET
095D 1895 :
095D 1896 : IF STATE NOECHO,15$ : BRANCH IF ECHO
54 0000'CF 9E 0961 1897 MOVAB W@TTY$A_CTRLU,R4 : ADDRESS MULTIECHO STRING
FF71 31 0966 1898 BRW 110$ : CONTINUE
0969 1899 :
FBBA 31 0969 1900 15$: BRW MOVEREADATA :
096C 1901 55$: CLR_STATE <WRAP,CURSOR> :
FF91 31 0974 1902 BRW CTRLU :
FF4E 31 0977 1903 57$: BRW CTRLR :
097A 1904 :
097A 1905 : NO DATA TO CONTROL-U
097A 1906 :
32 A4 B5 097A 1907 50$: TSTW TTY$W_RB_LINREST(R4) : ANY DATA AFTER THIS POINT
EA 12 097D 1908 BNEQ 15$ : YES THEN JUST RETURN
3C A4 B5 097F 1909 TSTW TTY$W_RB_TXTOFF(R4) : IS THERE ANY REASON TO DO THIS
E5 13 0982 1910 BEQL 15$ : NO THEN DON'T BOTHER
3C A4 B7 0984 1911 DECB TTY$W_RB_TXTOFF(R4) : REMOVE THE LAST CHARACTER
0637 30 0987 1912 BSBW FIND_BOL : THEN FIND THE BEGINNING
098A 1913 : OF THE PREVIOUS LINE
3C A4 B5 098A 1914 TSTW TTY$W_RB_TXTOFF(R4) : IS THERE ANY MORE DATA LEFT
E8 13 098D 1915 BEQL 57$ : NO THEN JUST MAKE SURE THE PROMPT
098F 1916 : ECHO'S
08 48 A5 18 E1 098F 1917 BBC #TT2$V_ANSICRT,UCB$L_DEVDEPND2(R5),55$ : REECHO ON NON ANSI TERMINALS
3A A4 B5 0994 1918 TSTW TTY$W_RB_CPZORG(R4) : WAS THIS AT THE RIGHT
D3 12 0997 1919 BNEQ 55$ : NO THEN USE THE OLD METHOD
3C A4 30 A4 A2 0999 1920 SUBW TTY$W_RB_LINOFF(R4),TTY$W_RB_TXTOFF(R4) : DELETE THE LINE
30 A4 B4 099E 1921 CLRW TTY$W_RB_LINOFF(R4) :
64 2C A4 D1 09A1 1922 CMPL TTY$B_RB_LIN(R4),TTY$B_RB_TXT(R4) : DOES A PROMPT NEED TO BE ECHOED
```

TTYCHARO
V04-000

L 8

- Terminal driver character output rout 16-SEP-1984 02:21:31 VAX/VMS Macro V04-00 Page 56
Post typeahead character action routines 5-SEP-1984 04:16:19 [TTDRVR.SRC]TTYCHARO.MAR;1 (41)

44	A4	0B	12	09A5	1923	BNEQ	608	:	NO THEN DON'T ECHO A PROMPT
		07	9B	09A7	1924	MOVZBW	#TTY\$K ER PRMECHO,TTY\$W	:	MODE(R4); SET UP TO ECHO THE PROMPT
		0B	A4	94	09AB	1925	CLRB	:	TTY\$B RB ECHLEN(R4) ; NO STRING TO OUTPUT
				09AE	1926	SET_STATE	EDITREAD	:	TELL THE REST OF THE SYSTEM ABOUT THE PROM
				09B2	1927				
		3B	A4	84	09B2	1928	608:	CLRW	TTY\$W RB CPZCUR(R4) ; CLEAR THE CURSOR POSITION
				09B5	1929	SET_STATE	SKIPCRF	:	SKIP THE FREE LINEFEED
54	00000000	'EF	9E	09B9	1930	MOVAB	TTY\$A ANSI_UPCEL,R4	:	GET THE STRING ADDRESS
	069D		31	09C0	1931	BRW	STRTMULTI	:	AND DELETE THIS LINE
				09C3	1932	.disable	lsb		

```
09C3 1934
09C3 1935 :++
09C3 1936 : DELCHAR - DELETE CHARACTER ROUTINE
09C3 1937 :
09C3 1938 : FUNCTIONAL DESCRIPTION:
09C3 1939 :
09C3 1940 : THIS ROUTINE DELETES THE LAST TYPED CHARACTER FROM THE READ BUFFER.
09C3 1941 : THEN IT SETS UP THE PROPER ECHO SEQUENCE FOR THE DELETED CHARACTER.
09C3 1942 :
09C3 1943 : INPUTS:
09C3 1944 :
09C3 1945 : R2 = ADDRESS OF TTY STATE VECTOR
09C3 1946 : R5 = UCB ADDRESS
09C3 1947 :
09C3 1948 : OUTPUTS:
09C3 1949 :
09C3 1950 : R2 = ADDRESS OF TTY STATE VECTOR
09C3 1951 : R5 = UCB ADDRESS
09C3 1952 : --
09C3 1953 : DELCHAR:
54 78 A5 D0 09C3 1954 : MOVL UCB$$_SVAPTE(R5),R4 : ADDRESS READ BUFFER BLOCK
09C7 1955 :
09C7 1956 : IF ESCAPE SEQUENCE IN PROGRESS THEN DELETE ENTIRE STRING
09C7 1957 :
09C7 1958 : IF STATE ESC,25$ : DELETE ESCAPE SEQUENCE
30 A4 B7 09C8 1959 : DECW TTY$$_RB_LINOFF(R4) : ANY DATA
30 19 09CE 1960 : BLSS 28$ : IF EQL THEN NO
51 30 A4 3C 09D0 1961 : MOVZWL TTY$$_RB_LINOFF(R4),R1 : GET THE POINTER
53 2C B441 9A 09D4 1962 : MOVZBL @TTY$$_RB_LIN(R4)[R1],R3 : GET THE CHARACTER
53 0D 91 09D9 1963 : CMPB #TTY$$_CR,R3 : IS THIS CHARACTER A RETURN
22 13 09DC 1964 : BEQL 28$ : YES THEN ADJUST THE LINE
3C A4 B7 09DE 1965 : DECW TTY$$_RB_TXTOFF(R4) : ADJUST POINTER
00 E1 09E1 1966 : IF NOT STATE NOECHO,5$ : BRANCH IF ECHO
6F 48 A5 09E5 1967 : BBC #TT2$$_LOCALECHO,- :
09EA 1968 : UCB$$_DEVDEPND2(R5),30$ : BRANCH IF NOT LOCAL ECHO
09EA 1969 :
09EA 1970 :
09EA 1971 : TEST FOR SPECIAL DELETES
09EA 1972 :
09 53 91 09EA 1973 5$: CMPB R3,#TTY$$_TAB : TAB?
03 13 09ED 1974 : BEQL 8$ : IF NEQ THEN OUTPUT JUST THE CHARACTER
00A3 31 09EF 1975 : BRW 55$
65 44 A5 0C E0 09F2 1976 8$: BBS #TT$$_SCOPE,UCB$$_DEVDEPEND(R5),40$; BR IF SCOPE
00DB 31 09F7 1977 : BRW 75$ : BR IF NOT SCOPE
09FA 1978 :
0144 30 09FA 1979 25$: BSBW DELESCAPE : DELETE ESCAPE SEQUENCE
FB26 31 09FD 1980 : BRW MOVEREADATA : AND GO BACK TO WORK
0A00 1981 :
30 A4 B4 0A00 1982 28$: CLRW TTY$$_RB_LINOFF(R4) : MAKE THE COUNT ZERO
32 A4 B5 0A03 1983 : TSTW TTY$$_RB_LINREST(R4) : is there nothing left?
51 12 0A06 1984 : BNEQ 30$ : NO THEN JUST RETURN
3C A4 B5 0A08 1985 : TSTW TTY$$_RB_TXTOFF(R4) : NO LINE WAS WRAPPED
4C 13 0A0B 1986 : BEQL 30$ : IF THIS COUNT IS ZERO
3C A4 B7 0A0D 1987 : DECW TTY$$_RB_TXTOFF(R4) : REMOVE THE LAST CHARACTER
05AE 30 0A10 1988 : BSBW FIND_BOL : YES THEN CALCULATE THE NEW BEGINNING
33 48 A5 18 E1 0A13 1989 : OF LINE
0A13 1990 : BBC #TT2$$_ANSICRT,UCB$$_DEVDEPND2(R5),120$; REECHO ON NON ANSI TERMINAL
```



```

      3A A4 B5 0A18 1991      TSTW      TTY$W_RB_CPZORG(R4)      ; WAS THIS AT THE RIGHT
      2E 12 0A1B 1992      BNEQ      120$      ; NO THEN USE THE OLD METHOD
      44 A4 01 9B 0A1D 1993      MOVZBW    #TTY$K_ER_CLRECHO,TTY$W_RB_MODE(R4); ECHO THE DATA
OB A4 30 A4 90 0A21 1994      MOVB      TTY$W_RB_CINOFF(R4),TTY$B_RB_ECHLEN(R4); AND THE STRING TO OUTPUT
14 A4 2C A4 D0 0A26 1995      MOVL      TTY$B_RB_LIN(R4),TTY$B_RB_ECHSTR(R4); AND THE ADDRESS
      64 2C A4 D1 0A2B 1996      CMPL      TTY$B_RB_LIN(R4),TTY$B_RB_TXT(R4); DOES A PROMPT NEED TO BE ECHOED
      04 12 0A2F 1997      BNEQ      110$      ; NO THEN DON'T ECHO A PROMPT
      44 A4 07 9B 0A31 1998      MOVZBW    #TTY$K_ER_PRMECHO,TTY$W_RB_MODE(R4); SET UP TO ECHO THE PROMPT
      0A35 1999
      0A35 2000 110$: SET STATE <SKIPCR LF,EDITREAD>      ; SKIP THE FREE LINEFEED
      38 A4 02 AE 0A3D 2001      MNEGW     #2,TTY$W_RB_CPZCUR(R4)      ; DON'T RESET CURSOR
54 00000000'EF 9E 0A41 2002      MOVAB     TTY$A_ANSI_OPCEL,R4      ; GET THE STRING ADDRESS
      0615 31 0A48 2003      BRW         STRTMULTI      ; AND DELETE THIS LINE
      0A4B 2004
      0A4B 2005 120$: SET STATE <EDITREAD>      ; RESTORE THE LINE TO IT'S FULL VIGOR
54 00000000'EF 9E 0A4F 2006      MOVAB     TTY$A_CTRLR,R4      ; GET A NEW LINE
      0607 31 0A56 2007      BRW         STRTMULTI      ; and return to the main loop
      FACA 31 0A59 2008      30$: BRW      MOVEREADATA      ; VECTOR TO CONTINUE
      0A5C 2009
      0A5C 2010
      0A5C 2011      ; DELETE A TAB ON A SCOPE
      0A5C 2012
      0A5C 2013
      00FC C5 B5 0A5C 2014 40$: TSTW      UCB$W_TT_CURSOR(R5)      ; Have we wrapped?
      47 13 0A60 2015      BEQL      60$      ; Branch to simulate CTRL-R, if yes.
      048A 30 0A62 2016      BSBW      BACK_TAB      ; GET THE TAB LENGTH
42 A5 00FC C5 B1 0A65 2017      CMPW      UCB$W_TT_CURSOR(R5),UCB$W_DEVBUSIZ(R5); At extreme right?
      11 1F 0A6B 2018      BLSSU     45$      ; Branch if within screen bounds.
      1C 1A 0A6D 2019      BGTRU     47$      ; Branch if beyond right edge.
      53 53 D6 0A6F 2020      INCL      R3      ; Adjust number of backspaces for
      FFFFFFFF 8F CA 0A71 2021      BICL      #^C7, R3      ; right-hand edge effects.
      41 13 0A78 2022      BEQL      66$      ; If no backspace, just delete at edge.
      00FC C5 B7 0A7A 2023      DECB     UCB$W_TT_CURSOR(R5)      ; Adjust cursor value for one less BS.
O0DC C5 07 53 A3 0A7E 2024 45$: SUBW3     R3,#7,UCB$W_TT_MULTILEN(R5); GET THE LENGTH CORRECTLY
54 0001'CF 9E 0A84 2025      MOVAB     W^TTY$A_DELCRTTAB+1,R4; Get address backspaces string.
      45 11 0A89 2026      BRB        70$      ; Go output them.
      0A8B 2027
      0A8B 2028 47$: SUBW      #7, R3      ; Adjust off-the-screen cursor
      00FC C5 53 A0 0A8E 2029      ADDW     R3,UCB$W_TT_CURSOR(R5)      ; position for deleted tab.
      26 11 0A93 2030      BRB        66$      ; Go do delete-at-edge.
      0A95 2031
      0A95 2032      ; NORMAL CHARACTER DELETE
      0A95 2033
      0A95 2034
      0A95 2035 55$:
      FO 8F 93 0A95 2036      BITB      #<TTY$M_CH_CTRL!TTY$M_CH_SPEC!TTY$M_CH_CTRL2!TTY$M_CH_CTRL3>,-
      0000'CF43 12 0A98 2037      W^TTY$A_TYPE[R3]      ; TEST FOR SPECIAL
      14 0A9C 2038      BNEQ      62$      ; NON PRINTING CHARACTER
      0A9E 2039
      32 44 A5 0C E1 0A9E 2040      BBC      #TTY$V_SCOPE,UCB$L_DEVDEPEND(R5),75$; BR IF NOT SCOPE
      0AA3 2041
      0AA3 2042      ; IF THE CURSOR IS AT THE LEFT MARGIN AND DATA IS PRESENT,
      0AA3 2043      ; FORCE A CONTROL R IN STEAD OF A BACKSPACE.
      0AA3 2044
      00FC C5 B5 0AA3 2045 59$: TSTW      UCB$W_TT_CURSOR(R5)      ; CURSOR AT LEFT MARGIN?
      12 12 0AA7 2046      BNEQ      66$      ; BR IF NORMAL BACKSPACE NEEDED
      0AA9 2047
```



```
0AA9 2048 : FORCE CONTROL R FOR RUBOUT RESPONSE
0AA9 2049 :
0AA9 2050 60$:
54 78 A5 D0 0AA9 2051 : MOVL UCBSL_SVAPTE(R5),R4 : GET READ PACKET ADDRESS
3C A4 B5 0AAD 2052 : TSTW TTY$W_RB_TXTOFF(R4) : DATA PRESENT?
03 12 0AB0 2053 : BNEQ 63$ : IF DATA PRESENT, ECHO IT IN NORMAL FASSION
FA71 31 0AB2 2054 62$: BRW MOVEREADATA : IF NO DATA THEN NO RESPONSE
FF93 31 0AB5 2055 63$: BRW 120$ : DATA TO ECHO THEN ECHO
0290 31 0AB8 2056 65$: BRW EDITREAD : START THE OUTPUT
0ABB 2057 :
0ABB 2058 : NORMAL BACKSPACE RUBOUT RESPONSE
0ABB 2059 :
42 A5 53 20 9A 0ABB 2060 66$: MOVZBL #TTY$C_BLANK,R3 : ASSUME SPACE IS FIRST CHARACTER
00FC C5 B1 0ABE 2061 : CMPW UCBSW_TT_CURSOR(R5),UCBSW_DEVBUSIZ(R5); AND EXTREME RIGHT?
74 1E 0AC4 2062 : BGEQU 100$ : IF YES THEN ONLY OUTPUT SPACE
0AC6 2063 :
0AC6 2064 : DELETE IS IN MID SCREEN
0AC6 2065 :
54 0000'CF 9E 0AC6 2066 : MOVAB WATTY$A_SPACEBACK,R4 : ADDRESS BACKSPACE STRING
00DC C5 84 9B 0ACB 2067 : MOVZBW (R4)+,UCBSW_TT_MULTILEN(R5); AND THEN IT'S LENGTH
53 08 9A 0ADO 2068 70$: MOVZBL #TTY$C_BS,R3 : START STRING WITH BACKSPACE
15 11 0AD3 2069 : BRB 90$ : START OUTPUT
SD 04 A2 01 E2 0AD5 2070 75$: BBSS #TTY$V_ST_DEL,4(R2),95$ : IF NOT FIRST TIME THEN OUTPUT CHARACTER
54 18 A5 9E 0ADA 2071 : MOVAB UCBSW_BUFQUO(R5),R4 : ADDRESS STRING SPACE
00DC C5 01 B0 0ADE 2072 : MOVW #1,UCBSW_TT_MULTILEN(R5) : SETUP THE LENGTH OF THE MULTIECHO
64 53 9B 0AE3 2073 : MOVZBW R3,(R4) : AND PUT THE CHARACTERS THERE
0AE6 2074 :
0AE6 2075 : OUTPUT '\' TO START OR END DELETE SEQUENCE
0AE6 2076 :
53 5C 8F 9A 0AE6 2077 80$: MOVZBL #A@\\@,R3 : FOR FIRST TIME OUTPUT '\'
00D8 C5 54 D0 0AEA 2078 90$: MOVL R4,UCBSL_TT_MULTI(R5) : ADDRESS STRING
0AEF 2079 : SET STATE MULTI
54 78 A5 D0 0AF3 2080 : MOVL UCBSL_SVAPTE(R5),R4 : RESTORE THE READ PACKET ADDRESS
32 A4 B5 0AF7 2081 : TSTW TTY$W_RB_LINREST(R4) : ANY CHARACTERS TO THE RIGHT
3B 13 0AFA 2082 : BEQL 95$ : NO THEN HANDLE NORMALY
0AFC 2083 :
3F BB 0AFC 2084 : PUSHR #M<R0,R1,R2,R3,R4,R5> : SAVE NECESSARY STATE
0AFE 2085 : IF STATE NOECHO,92$
0B02 2086 : SET STATE <EDITREAD> : SET THE EDIT ECHOING STATE
14 A4 51 30 A4 3C 0B06 2087 92$: MOVZWL TTY$W_RB_LINOFF(R4),R1 : GET THE OFFSET TO THIS CHARACTER
2C B441 9E 0B0A 2088 : MOVAB @TTY$C_RB_LIN(R4)[R1],TTY$C_RB_ECHSTR(R4); SETUP THE ADDRESS TO STAR
53 14 B4 9A 0B10 2089 : MOVZBL @TTY$C_RB_ECHSTR(R4),R3 : GET THE CHARACTER BACK
44 A4 01 9B 0B14 2090 : MOVZBW #TTY$K_ER_CLRECHO,TTY$W_RB_MODE(R4); CLEAR TO THE END OF LINE THEN E
OB A4 32 A4 90 0B18 2091 : MOVW TTY$W_RB_LINREST(R4),TTY$B_RB_ECHLEN(R4); AND THE LENGTH
51 01 CE 0B1D 2092 : MNEGL #1,R1 : Setup to find backward size
042F 30 0B20 2093 : BSBW CHAR_SIZE : FIND THE SIZE OF THIS CHARACTER
38 A4 00FC C5 53 A3 0B23 2094 : SUBW3 R3,UCBSW_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4) : SUBTRACT OUT THE L
52 01 9A 0B2A 2095 : MOVZBL #1,R2 : SETUP R2 AS 1 CHARACTER
32 A4 28 0B2D 2096 : MOVCL TTY$W_RB_LINREST(R4) -
14 B4 14 B442 0B30 2097 : @TTY$C_RB_ECHSTR(R4)[R2],@TTY$C_RB_ECHSTR(R4); COMPRESS THE STRING
3F BA 0B35 2098 : POPR #M<R0,R1,R2,R3,R4,R5> : SAVE NECESSARY STATE
0B37 2099 :
F64D 31 0B37 2100 95$: BRW FORMAT_CHAR : START UP OUTPUT
00FC C5 B7 0B3A 2101 100$: DECW UCBSW_TT_CURSOR(R5) : ADJUST CURSOR
F607 31 0B3E 2103 : BRW OUTPUTANDWAIT : VECTOR FOR SPACE ONLY RESPONSE
```

- Terminal driver character output routines 16-SEP-1984 02:21:31 VAX/VMS Macro V04-00 Page 60
Post typeahead character action routines 5-SEP-1984 04:16:19 [T:DRV.RSRC]TTYCHARO.MAR;1 (45)

```

OB41 2105
OB41 2106 :++
OB41 2107 : DELETE AN ESCAPE SEQUENCE IN PROGRESS
OB41 2108 :
OB41 2109 : THIS ROUTINE IS USED BY THE CONTROL U AND DELETE LOGIC TO RESET
OB41 2110 : AND DELETE THE ESCAPE SEQUENCE IN PROGRESS
OB41 2111 :
OB41 2112 DELESCAPE:
3C 51 58 A5 D0 OB41 2113 : DELESCAPE: : DELETE CURRENT ESCAPE SEQUENCE
3C A4 3A A1 A2 OB45 2114 : MOVL UCBSL_IRP(R5),R1 : GET ADDRESS OF PACKET.
3C A4 3B A1 D4 OB4A 2115 : SUBW IRPSL_MEDIA+2(R1),TTY$W-RB : ADJUST TRANSFER SIZE FOR SEQU
OB4D 2116 : CLR IRPSL_MEDIA(R1) : ESCAPE LENGTH
05 OB53 2117 : CLR STATE <ESC,BADESC> : CLEAR ESCAPE
10$ : RSB : RETURN

```

```
OB54 2119 :++
OB54 2120 : DELETE_WORD
OB54 2121 :
OB54 2122 : DESCRIPTION:
OB54 2123 : DELETE WORD TO THE LEFT OF THE CURSOR
OB54 2124 :--
FE6C 31 OB54 2125 GODEL: BRW DELCHAR : ACT AS IF A DELETE WAS TYPED
F73F 31 OB57 2126 DROP_CHAR:BRW DROP : DROP THE CHARACTER
OB5A 2127 DELETE_WORD:
53 30 A4 3C OB5A 2128 MOVZWL TTY$W_RB_LINOFF(R4),R3 : GET THE OFFSET
53 D7 OB5E 2129 DECL R3 : SUBTRACT 1
F5 19 OB60 2130 BLSS DROP_CHAR : no characters then handle suchly
53 2C B443 9A OB62 2131 MOVZBL @TTY$LB_LIN(R4)[R3],R3 : GET THE CHARACTER AT THIS POINT
E5 00000000'EF 53 E0 OB67 2132 BBS R3,TTY$A_PREFIX,GODEL: IF : THIS CHARACTER
OB6F 2133 : IS A PREFIX THEN DELETE ONLY IT
OB6F 2134 PUSHF #M<R0,R1,R2,R4,R5>
50 30 A4 3C OB71 2135 MOVZWL TTY$W_RB_LINOFF(R4),R0 : GET THE CURRENT POSITION
50 DD OB75 2136 PUSHF R0 : SAVE THE OFFSET
51 01 CE OB77 2137 MNEGL #1,R1 : WE ARE MOVING BACKWARDS
52 D4 OB7A 2138 CLRL R2 : CLEAN OUT THE CHARACTER POSITION COUNTER
50 D7 OB7C 2139 10$: DECL R0 : MOVE ONE CHARACTER TO THE LEFT
34 19 OB7E 2140 BLSS 30$ : WHEN DONE SAY SO
53 2C B440 9A OB80 2141 MOVZBL @TTY$LB_LIN(R4)[R0],R3 : GET A CHARACTER
OC 00000000'EF 53 E1 OB85 2142 BBC R3,TTY$A_WORDTERM,20$ : SKIP OVER TERMINATORS
30 A4 50 B0 OB8D 2143 MOVW R0,TTY$W_RB_LINOFF(R4) : UPDATE LINOFF TO MAKE TAB DELETION WORK
52 03BE 30 OB91 2144 BSBW CHAR_SIZE : GET THE LENGTH OF THIS CHARACTER
52 53 C0 OB94 2145 ADDL R3,R2 : UPDATE CURSOR POSITION
E3 11 OB97 2146 BRB 10$ : AND CONTINUE
OB99 2147
30 A4 50 B0 OB99 2148 20$: MOVW R0,TTY$W_RB_LINOFF(R4) : UPDATE LINOFF TO MAKE TAB DELETION WORK
52 03BE 30 OB9D 2149 BSBW CHAR_SIZE : GET THE LENGTH OF THIS CHARACTER
52 53 C0 OBA0 2150 ADDL R3,R2 : UPDATE THE POSITION
50 D7 OBA3 2151 DECL R0 : KEEP MOVING
OD 19 OBA5 2152 BLSS 30$ : UNTILL THE END
53 2C B440 9A OBA7 2153 MOVZBL @TTY$LB_LIN(R4)[R0],R3 : GET ANOTHER CHARACTER
E5 00000000'EF 53 E1 OBAC 2154 BBC R3,TTY$A_WORDTERM,20$ : KEEP MOVING UNTILL NOT A TERMINATOR
OBBA 2155
30$: INCL R0 : GO FORWARD A CHARACTER
OBBA 2156 MOVW R2,UCB$W_TT_BSLEN(R5) : ECHO THE RIGHT NUMBER OF BACKSPACES
38 A4 0100 C5 52 B0 OBBA 2157 SUBW3 R2,UCB$W_TT_CURSOR(R5),TTY$W_RB_CP2CUR(R4) : SETUP THE CURSOR POSITION
00FC C5 52 A3 OBBA 2158 POPL R1 : GET THE CURRENT POSITION
52 51 50 A3 OBC5 2159 SUBW3 R0,R1,R2 : GET THE NUMBER OF CHARACTERS DELETED
3C A4 52 A2 OBC9 2161 SUBW R2,TTY$W_RB_TXTOFF(R4) : AND REMOVE THESE CHARACTERS FROM THE COUNT
14 A4 2C B440 9E OBCD 2162 MOVAB @TTY$LB_LIN(R4)[R0],TTY$LB_ECHSTR(R4) : GET THE ADDRESS FOR UPDAT
32 A4 28 OBD3 2163 MOVCL TTY$W_RB_LINREST(R4) : DELETE THE WORD
14 B4 2C B441 OBD6 2164 @TTY$LB_LIN(R4)[R1],@TTY$LB_ECHSTR(R4) :
37 BA OBD8 2165 POPR #M<R0,R1,R2,R4,R5> : RESTORE THE STATE
OBDD 2166 IF STATE NOECHO,40$
OBE1 2167 SET STATE <BACKSPACE,EDITREAD> : BACKSPACE OVER THE WORD THEN ECHO EVERYTHI
44 A4 01 98 OBE6 2168 MOVZBL #TTY$K_ER_CLRCHO,TTY$W_RB_MODE(R4) : SETUP THE MODE FOR EDITING
OB A4 32 A4 90 OBEA 2169 MOVW TTY$W_RB_LINREST(R4),TTY$B_RB_ECHLEN(R4) : AND THE LENGTH TO ECHO
F413 31 OBEF 2170 40$: BRW GETNEXTCHAR : GET THE NEXT CHARACTER
OBF2 2171
```

```

OBF2 2173 :++
OBF2 2174 : Escape_Char
OBF2 2175 :
OBF2 2176 : Possible escape character found, could be an altmode for
OBF2 2177 : lower case terminals or could be a CSI for new eight bit terminals.
OBF2 2178 : All cases must be handled appropriately.
OBF2 2179 :--
OBF2 2180 ESCAPE_CHAR:
53 7D 8F 91 OBF2 2181 CMPB #TTYSC_LOWESC1,R3 : ALT MODE?
13 OBF2 2182 BEQL 47$ : YES
53 7E 8F 91 OBF2 2183 CMPB #TTYSC_LOWESC2,R3 : ALT MODE?
12 OBF2 2184 BNEQ 52$ : YES
39 44 A5 07 E0 OBF2 2185 47$: BBS #TT$V_LOWER,UCB$L_DEVDEPEND(R5),51$; LOWER CASE TERMINAL CONTINUE
53 1B 9A OC03 2186 MOVZBL #TTYSC_ESCAPE,R3 : CHANGE CHARACTER TO ESCAPE
OC06 2187 52$:
OC06 2188
OC06 2189 : PROCESS ESCAPE SEQUENCES AND ESCAPES
OC06 2190 :
OC06 2191 : IF NOT_STATE <ESCAPE,EDITING>,55$; NOT AN ESCAPE TERMINAL THEN EXIT
0103 C5 F88B 30 OC0E 2192 BSBW ESCINIT : HANDLE THE ESCAPE PREFIXES
51 48 A4 B4 OC11 2193 MOVW R1,UCB$B_TT_ESC(R5)
51 58 A5 D0 OC16 2194 CLRW TTY$W_RB_ESC_TKN(R4)
38 A1 53 B0 OC19 2195 MOVL UCB$L_IRP(R5),R1 : GET THE IRP ADDRESS
51 3A A1 B6 OC21 2196 MOVW R3,IRP$L_MEDIA(R1) : SET TERMINATOR CHARACTER
00 B441 3C A4 3C OC24 2197 INCW IRP$L_MEDIA+2(R1) : START TERMINATOR SIZE
3C A4 B6 OC28 2198 MOVZWL TTY$W_RB_TXTOFF(R4),R1 : GET THE PLACE TO STORE THE ESCAPE
OC2D 2199 MOVW #TTYSC_ESCAPE,@TTY$L_RB_TXT(R4)[R1]; AND PLACE THE CHARACTER IN THE
OC30 2200 INCW TTY$W_RB_TXTOFF(R4) : AND INCREMENT THE WORLD
OC34 2201 IF_STATE ESC,60$
OC39 2202 SET_STATE ESC
F957 31 OC39 2203 BRW CHECK_BUFFER : BEGIN ESCAPE SEQUENCE
OC3C 2204 : INSERT THE ESCAPE AT THE END OF THE BUFFER
OC3C 2205 51$:
OC3C 2206 IF_NOT_STATE ESC,55$
FA1C 31 OC40 2207 BRW ESCINPROG : ESCAPE IN PROGRSS THEN CONTINUE
OC43 2208 55$:
F92B 31 OC43 2209 BRW INSERT_CHAR : ELSE INSERT THE CHARACTER
OC46 2210 :
OC46 2211 : ALREADY IN ESCAPE MODE THEN MAKE THIS BAD
OC46 2212 :
OC46 2213 60$: SET_STATE <BADESC,EOL>
F3B4 31 OC4E 2214 BRW GETNEXTCHAR :
```



```
OC51 2216
OC51 2217 :++
OC51 2218 : FORWARD CHAR
OC51 2219 : MOVE THE CURSOR FORWARD 1 CHARACTER POSITION.
OC51 2220 :--
OC51 2221 FORWARD_CHAR:
38 A4 51 30 A4 3C OC51 2222 MOVZWL TTY$W_RB_LINOFF(R4),R1 : GET THE OFFSET TO THE CHARACTER
53 53 2C B441 9A OC55 2223 MOVZBL @TTY$C_RB_LIN(R4)[R1],R3 : GET THE CHARACTER
32 A4 B7 OC5A 2224 DECW TTY$W_RB_LINREST(R4) : FORWARD A CHARACTER
27 19 OC5D 2225 BLSS 50$ : AT THE END THEN DON'T ALLOW IT
51 D4 OC5F 2226 CLRL R1 : GOING FORWARD
02EE 30 OC61 2227 BSBW CHAR_SIZE : CHECK THE SIZE OF THE CHARACTER
OUFC C5 53 A1 OC64 2228 ADDW3 R3,UCBSW_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4) : UPDATE THE CURSOR POSITI
51 53 30 A4 3C OC6B 2229 MOVZWL TTY$W_RB_LINOFF(R4),R1 : GET THE OFFSET TO THE CHARACTER
53 53 2C B441 9A OC6F 2230 MOVZBL @TTY$C_RB_LIN(R4)[R1],R3 : GET THE CHARACTER
30 A4 B6 OC74 2231 INCW TTY$W_RB_LINOFF(R4) : UPDATE THE REST POINTER
53 09 91 OC77 2232 CMPB #TTY$C_TAB,R3 : ARE WE FORWARDING OVER A TAB?
03 12 OC7A 2233 BNEQ 10$ : NO THEN CONTINUE
040E 30 OC7C 2234 BSBW TABRIGHT : YES THEN CHECK FOR A TAB TO THE RIGHT
F501 31 OC7F 2235 10$: IF STATE NOECHO,GODROP
OC83 2236 BRW FORMAT_CHAR : THEN OUTPUT THE CHARACTER TO GO FOWARD
OC86 2237
32 A4 B4 OC86 2238 50$: CLRW TTY$W_RB_LINREST(R4) : RESET THE POINTER
F60D 31 OC89 2239 GODROP: BRW DROP : AND DROP THE CHARACTER
```

```
OC8C 2241 :++  
OC8C 2242 : MOVE_BOL  
OC8C 2243 :  
OC8C 2244 : DESCRIPTION:  
OC8C 2245 : GO TO THE BEGINNING OF THE LINE.  
OC8C 2246 :--  
OC8C 2247 : MOVE_BOL:  
3A A4 B5 OC8C 2248 TSTW TTY$W_RB_CPZORG(R4) : START AT ZERO?  
03 13 OC8F 2249 BEQL 10$ : YES THEN NO SPECAIL ACTION  
0484 30 OC91 2250 BSBW ZERO CPZORG : ZERO ORG CURSOR IF WE HAVE WRAPPED  
32 A4 30 A4 A0 OC94 2251 10$: ADDW TTY$W_RB_LINOFF(R4),TTY$W_RB_LINREST(R4); MOVE THE CHARACTER COUNTS  
30 A4 B4 OC99 2252 CLRW TTY$W_RB_LINOFF(R4) : AND CLEAR THE NUMBER OF CHARACTERS WE HAVE  
03EE 30 OC9C 2253 BSBW TABRIGHT : CHECK FOR TABS TO THE RIGHT  
64 2C A4 D1 OC9F 2254 CMPL TTY$L_RB_LIN(R4),TTY$L_RB_TXT(R4); ARE WE ON THE FIRST LINE  
0C 12 OCA3 2255 BNEQ 20$ : NO THEN JUST DROP IT  
0B A4 01 8E OCA5 2256 MNEGB #1,TTY$B_RB_ECHLEN(R4) : CLEAN OUT THE STRING TO ECHO  
44 A4 07 9B OCA9 2257 MOVZBW #TTY$K_ER_PRMECHO,TTY$W_RB_MODE(R4); SETUP THE MODE  
OCAD 2258 SET STATE <EDITREAD>  
3A A4 B5 OC81 2259 20$: TSTW TTY$W_RB_CPZORG(R4) : START AT ZERO?  
1D 12 OC84 2260 BNEQ 50$ : YES THEN NO SPECAIL ACTION  
0008 C5 00000001'EF 9E OC86 2261 MOVAB TTY$A_ANSI_DEOL+1,UCB$L_IT_MULTI(R5); GET THE ADDRESS OF A <CR>  
00DC C5 01 D0 OC8F 2262 MOVL #1,UCB$W_IT_MULTILEN(R5); AND ONLY OUTPUT THE <CR>  
OCC4 2263 SET STATE <MULTI,SKIPLF,SKIPCRLF>; SETUP TO DO A MULTIECHO  
38 A4 B4 OCCE 2264 CLRW TTY$W_RB_CPZCUR(R4) : MOVE TO THE BEGINNING WHEN DONE  
B6 11 OCD1 2265 BRB GODROP : THEN DISPATCH  
OCD3 2266 :  
OCD3 2267 : IF THE ORIGIN IS NON-ZERO THEN MOVE THE CURSOR TO THE ORIGIN POSITION  
OCD3 2268 :  
38 A4 3A A4 B0 OCD3 2269 50$: MOVW TTY$W_RB_CPZORG(R4),TTY$W_RB_CPZCUR(R4); SETUP TO MOVE BACK  
03CE 31 OCD8 2270 BRW UPDATE_CURSOR
```

				OCDB	2272	::++	
				OCDB	2273	::	MOVE_EOL
				OCDB	2274	::	
				OCDB	2275	::	DESCRIPTION:
				OCDB	2276	::	MOVE TO THE END OF THE LINE.
				OCDB	2277	::--	
				OCDB	2278	::	MOVE_EOL:
08 A4	32 A4	90	OCDB	2279	MOV	TTY\$W_RB_LINREST(R4), TTY\$B_RB_ECHLEN(R4);	GET THE LENGTH TO ECHO
51	30 A4	3C	OCE0	2280	MOVZWL	TTY\$W_RB_LINOFF(R4), R1	; GET THE OFFSET OF THIS CHARACTER
14 A4	2C B441	9E	OCE4	2281	MOVAB	@TTY\$C_RB_LIN(R4)[R1], TTY\$B_RB_ECHSTR(R4);	SETUP THE ADDRESS ALSO
44 A4	02	9B	OCEA	2282	MOVZBW	#TTY\$K_ER_ECHLINE, TTY\$W_RB_MODE(R4);	AND SETUP THE MODE
			OCEE	2283	CLR_STATE	TABRIGHT	; NO MORE CHARACTERS TO THE RIGHT SO
			OCF3	2284			; NO TABS EITHER
			OCF3	2285	IF STATE	NOECHO, 10\$; NO ECHO THEN DON'T ECHO
			OCF7	2286	SET STATE	EDITREAD	; THEN THE STATE
30 A4	32 A4	A0	OCF8	2287	10\$: ADD	TTY\$W_RB_LINREST(R4), TTY\$W_RB_LINOFF(R4);	FIX THE COUNTS
	32 A4	B4	OD00	2288	CLRW	TTY\$W_RB_LINREST(R4)	; GET THE NEXT CHARACTER
	F2FF	31	OD03	2289	BRW	GETNEXTCHAR	

```

OD06 2291 :++
OD06 2292 : QUOTING
OD06 2293 :
OD06 2294 : DESCRIPTION:
OD06 2295 : INITIATE QUOTING SEQUENCE
OD06 2296 : --
OD06 2297 QUOTING:
OD06 2298 SET_STATE QUOTING ;SET THE STATE QUOTE
F2F7 31 OD0B 2299 BRW GETNEXTCHAR ; AND CONTINUE ON

```

			ODOE	2301	::++	
			ODOE	2302	:: RECALL	
			ODOE	2303	::	
			ODOE	2304	:: DESCRIPTION:	
			ODOE	2305	:: SETUP THE NECESSARY STATE TO RECALL THE LAST COMMAND.	
			ODOE	2306	::--	
			ODOE	2307	RECALL:	
28 20	A4 10	E0	ODOE	2308	BBS	#TRMSV TM NORECALL, TTY\$LB MOD(R4), 20\$
54	00E4 C5	D0	OD13	2309	MOVL	UCB\$LB_TT_TYPAHD(R5), R4 ; GET THE ADDRESS OF THE TYPEAHEAD BUFFER
	14 A4	B5	OD18	2310	TSTW	TTY\$W_TA_RCLSIZ(R4) ; CHECK THE SIZE OF THE BUFFER
	1B	13	OD1B	2311	BEQL	10\$; DON'T DO ANYTHING IF NO CHARACTERS
	0E A4	B4	OD1D	2312	CLRW	TTY\$W_TA_RCOFF(R4) ; CLEAN OUT THE OFFSET TO THE STRING
			OD20	2313	SET STATE	<RECALL, EDITREAD> ; TELL THE DISPATCHER THAT WE ARE RECALLING
54	78 A5	D0	OD24	2314	MOVE	UCB\$LB_SVAPTE(R5), R4 ; GET THE PACKET ADDRESS
	3C A4	B4	OD28	2315	CLRW	TTY\$W_RB_TXTOFF(R4) ; NO MORE TEXT
	30 A4	B4	OD2B	2316	CLRW	TTY\$W_RB_LINOFF(R4) ; CLEAR THE LINE
	32 A4	B4	OD2E	2317	CLRW	TTY\$W_RB_LINREST(R4) ; EVERYTHING ELSE
2C A4	64	D0	OD31	2318	MOVL	TTY\$LB_RB_TXT(R4), TTY\$LB_RB_LIN(R4)
	FB90	31	OD35	2319	BRW	CTRLR ; THEN CONTROL-R THE READ
	F2CA	31	OD38	2320	BRW	GETNEXTCHAR
	F833	31	OD3B	2321	BRW	INSERT_CHAR
					10\$:	
					20\$:	


```

OD3E 2323 :++
OD3E 2324 : TOGGELINOV
OD3E 2325 :
OD3E 2326 : Description
OD3E 2327 : toggle the insert/overstrike mode flag
OD3E 2328 :--
OD3E 2329 TOGGELINOV:
05 04 A2 17 E4 OD3E 2330 BBSC #TTYSV ST OVERSTRIKE,4(R2),10$; TOGGEL THE OVERSTRIKE BIT
OD43 2331 SET_STATE OVERSTRIKE
F2BA 31 OD48 2332 10$: BRW GETNEXTCHAR

```

```
.SBTTL EDITREAD - READ EDITING STATE

++
EDITREAD - STATE TO ALLOW A READ EDITING SEQUENCE

FUNCTIONAL DESCRIPTION:
    WHEN ECHOING PARTS OF READS IT IS NECESSARY TO ECHO SEVERAL
    STRINGS, THIS ROUTINE FILLS THAT NEED

INPUTS:
    R2 = ADDRESS OF THE UNIT STATE VECTOR
    R5 = UCB ADDRESS

OUTPUTS:
    R2 = ADDRESS OF THE UNIT STATE VECTOR
    R4 = ADDRESS OF THE READ PACKET
    R5 = UCB ADDRESS

--
EDITREAD:
    MOVL    UCBSL_SVAPTE(R5),R4      ; READ EDITING
    CASE    TTY$W_RB_MODE(R4),TYPE=W,<- ; GET THE READ PACKET ADDRESS

    NORMAL,-
    CLRECHO,-                        ; ECHO WITH CLEAR BIT SET
    ECHLINE,-                        ; JUST ECHO WHAT IS SPECIFIED
    UPDCURSOR,-                      ; UPDATE THE CURSOR POSITION
    EXITING,-                        ; SETUP TO EXIT.
    MOVECURSOR,-                    ; ALLOW CURSOR TO BE UPDATED
    CLRREST,-                       ; CLEAR THE REST OF THE LINE
    PRMECHO,-                        ; ECHO PROMPT THEN CONTINUE
    PRMECHO1,-                      ; CONTINUING STATE FOR PROMPTING
    AESECHO,-                       ; ECHO AES STRING AND NOSTRING
    RVECHO,-                         ; echo in the readverify way
    SINCEOL>                        ; SIMULATE CLEAR TO END OF LINE

NORMAL:
    IF STATE <CTRLR,RDVERIFY>,120$   ; HAVE WE COMPLETED THE CONTROL-R
    TSTW    TTY$W_RB_CPZORG(R4)      ; START AT ZERO?
    BEQL    25$                      ; YES THEN NO SPECAIL ACTION
    TSTW    UCBSW_TT_CURSOR(R5)      ; AND WE ARE NOW AT ZERO
    BNEQ    25$                      ; YES THEN WE SHOULD FIXUP THE LINE
    CLRW    TTY$W_RB_CPZORG(R4)      ; DO THE FIXING
    BSBW    FIND_BOL                 ; FIND THE BEGINNING OF THE LAST LINE
    CLRW    UCBSW_TT_CURSOR(R5)      ; FIX THE CURSOR (CHANGED DURING FIND_BOL)
    IF NOT STATE SKIPCRLF,119$        ; NO THEN ARE WE DOING A CR <CEL>
    SET STATE <SKIPLF>                ; YES THEN SKIP A LEADING LINEFEED AND CONTI
    MOVZBW  #TTY$K_ER_CLRREST,TTY$W_RB_MODE(R4); THEN CLEAR THE REST
    TSTW    TTY$W_RB_CINREST(R4)      ; CURSOR VALID?
    BNEQ    119$                     ; YES THEN USE IT
    MNEGW   #1,TTY$W_RB_CPZCUR(R4)    ; ELSE USE THE END POSITION
    SET STATE <CTRLR,EDITREAD>        ; INDICATE THAT WE ARE OUTPUTTING THE
    ADDQ3   TTY$W_RB_TXTOFF(R4),TTY$W_RB_PRMLEN(R4),- ; GET THE NUMBER OF CHARACTERS TO OUTPUT
    BEQL    120$                     ; IF NONE THEN DON'T BOTHER.

54 78 A5 D0
    3A A4 B5
    10 13
    00FC C5 B5
    0A 12
    3A A4 B4
    0238 30
    00FC C5 B4
    44 A4 06 9B
    32 A4 B5
    04 12
    38 A4 01 AE
    34 A4 3C A4 A1
    51 0DAF
    07 13 ODB0
```

```
54  4A A4  9E ODB2 2392      MOVAB  TTY$A_RB_PRM(R4),R4      : GET THE ADDRESS TO START AT
    02B5  31 ODB6 2393      BRW      STRTMOLTI_1      : AND START THE STRING ECHOING.
    ODB9 2394      :
    ODB9 2395      : END OF CONTROL R
    ODB9 2396      :
51  14 A4  D0 ODB9 2397 120$: MOVL  TTY$A_RB_ECHSTR(R4),R1 : CHECK FOR A CALLBACK
    1D    12 ODBD 2398      BNEQ    CALLBACK
    ODBF 2399      MOVIFNEC:
    32 A4  B5 ODBF 2400      TSTW    TTY$W_RB_LINREST(R4) : ANY MORE CHARACTERS ON THE LINE
    03    13 ODC2 2401      BEQLU   EXITING : NO THEN EXIT
    0083  31 ODC4 2402      BRW      MOVECURSOR : YES THEN MOVE THE CURSOR
    44 A4  B4 ODC7 2403      EXITING:CLRW  TTY$W_RB_MODE(R4) : CLEAN OUT THE EDITING MODE
    14 A4  D4 ODCA 2404      CLRL    TTY$A_RB_ECHSTR(R4)
    ODCD 2405      CLR_STATE <EDITREAD,CTRLR,CTRLO,SKIPCRLF,TABEXPAND>
    F229  31 ODD9 2406      BRW      GETNEXTCHAR
    ODDC 2407      :
    ODDC 2408      : IT MAY BE NECESSARY TO CALL THE EDITING CODE BACK TO DO SOME WORK OPERATION
    ODDC 2409      : THAT REQUIRED A ZERO ORIGINIAL CURSOR POSITION
    ODDC 2410      :
    44 A4  B4 ODDC 2411      CALLBACK:CLRW  TTY$W_RB_MODE(R4) : CLEAN OUT THE EDITING MODE
    14 A4  D4 ODDF 2412      CLRL    TTY$A_RB_ECHSTR(R4)
    ODE2 2413      CLR_STATE <EDITREAD,CTRLR,CTRLO,SKIPCRLF,TABEXPAND>
    3A A4  B5 ODEE 2414      TSTW    TTY$W_RB_CPZORG(R4) : MAKE SURE WE WON'T LOOP
    D4    12 ODF1 2415      BNEQ    EXITING : IF NOT SUCCESSFULL IN CLEARING
    ODF3 2416      : THE ORIGIN THEN DROP THE CHARACTER
    61    17 ODF3 2417      JMP      (R1)
```

TTYCHAR0
V04-000

N 9
- Terminal driver character output rout 16-SEP-1984 02:21:31 VAX/VMS Macro V04-00 Page 71
EDITREAD - READ EDITING STATE 5-SEP-1984 04:16:19 [TTDRVR.SRC]TTYCHAR0.MAR;1 (57)

```

      ODF5 2419 :
      ODF5 2420 : ALTERNATE ECHO STRING.
      ODF5 2421 :
44 A4 04 B0 ODF5 2422 AESECHO: MOVW #TTYSK_ER_EXITING,TTYSW_RB,MODE(R4); SETUP TO EXIT
51 28 A4 3C ODF9 2423 MOVZWL TTYSW_RB_AESLEN(R4),R1 ; THE LENGTH OF THE STRING TO ECHO
54 24 A4 D0 ODFD 2424 MOVL TTYSL_RB_AES(R4),R4 ; AND THE ADDRESS
      026A 31 OE01 2425 BRW STRTMULTI_1 ; NOW MULTIECHO
```

				OE04	2427	:	
				OE04	2428	:	Echo and clear to end of the Line
				OE04	2429	:	
				OE04	2430	:	CLRECHO:
				OE04	2431	:	SET STATE <TABEXPAND>
44	A4	06	9B	OE08	2432	:	MOVZBW #TTYSK_ER CLRREST,TTYSW_RB MODE(R4); THEN MOVE THE CURSOR
51		0B A4	9A	OE0C	2433	:	COMECHO:MOVZBL TTYSB_RB_ECHLEN(R4),R1 - ; SETUP THE NUMBER OF CHARACTERS
54		14 A4	D0	OE10	2434	:	MOVL TTYSL_RB_ECHSTR(R4),R4 ; SETUP FOR MULTIECHO
		0257	31	OE14	2435	:	BRW STRTMULTI_1 ; AND START THE MULTIECHOING


```
OE17 2437 :  
OE17 2438 : Clear to the end of the line  
OE17 2439 :  
OE17 2440 CLRREST:  
44 A4 05 9B OE17 2441 MOVZBW #TTY$K_ER_MOVECURSOR,TTY$W_RB_MODE(R4);  
38 A4 38 A4 06 B5 OE1B 2442 TSTW TTY$W_RB_CPZCUR(R4) ; NEGITIVE CUR MEANS DON'T UPDATE  
38 A4 00FC C5 B0 OE1E 2443 BGEQ 20$  
42 A5 00FC C5 B1 OE20 2444 MOVW UCBSW-TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); UPDATE THE CURRENT  
1C 13 OE26 2445 20$: CMPW UCBSW-TT_CURSOR(R5),UCBSW_DEVBUFSIZ(R5); ANYTHING TO CLEAR  
OE2C 2446 BEQL MOVECURSOR ; NOTHING TO CLEAR THEN JUST MOVE THE  
OE2E 2447 ; CURSOR  
0A 48 A5 18 E1 OE2E 2448 BBC #TT2$V_ANSICRT,UCBSL_DEVDEPND2(R5),30$; IS IT AN ANSI CRT  
54 00000000'EF 9E OE33 2449 MOVAB TTY$A_ANSICEL,R4 ; GET A CLEAR TO END OF LINE SEQUENCE  
0223 31 OE3A 2450 BRW STRTMULTI ; AND MULTIECHO IT OUT  
OE3D 2451 ; for non-ansi terminals  
44 A4 0B 9B OE3D 2452 30$: MOVZBW #TTY$K_ER_SIMCEOL,TTY$W_RB_MODE(R4); TELL IT TO COME BACK AND SIMULA  
0075 31 OE41 2453 BRW SIMCEOL
```

```

        OE44 2455 :
        OE44 2456 : echo the given data then exit
        OE44 2457 :
44 A4      04 9B OE44 2458 ECHLINE:MOVZBW #TTYSK_ER_EXITING,TTYSW_RB_MODE(R4); EXIT AFTER ECHOING
        C2 11 OE48 2459 BRB COMECHO ; AND GO TO THE COMMON CODE
    
```

```

      OE4A 2461 :
      OE4A 2462 : cause the cursor to be moved to it's original position
      OE4A 2463 :
      OE4A 2464 MOVECURSOR:
44 A4 04 9B OE4A 2465 MOVZBW #TTY$K_ER_EXITING,TTY$W_RB_MODE(R4); SETUP TO EXIT
      0258 31 OE4E 2466 BRW UPDATE_CURSOR-; AND UPDATE THE CURSOR
  
```

```
OE51 2468 :  
OE51 2469 : Echo a prompt and a given amount of data then move the cursor to  
OE51 2470 : the specified position. This handles the no prompt case correctly.  
OE51 2471 :  
OE51 2472 PRMECHO:  
51 34 A4 B0 OE51 2473 MOVW TTY$W_RB_PRMLEN(R4),R1 ; GET THE LENGTH OF THE PROMPT  
11 13 OE55 2474 BEQL 40$ ; NO PROMPT THEN JUST ECHO SPECIFIED STRING  
44 A4 08 9B OE57 2475 MOVZBW #TTY$K_ER_PRMECHO1,TTY$W_RB_MODE(R4); SET THE NEXT STATE  
54 4A A4 9E OE5B 2476 MOVAB TTY$A_RB_PRM(R4),R4 ; GET THE PROMPT'S ADDRESS  
OE5F 2477 SET_STATE <SKIPCRLF,SKIPLF> ; SKIP THE FREE LINEFEEDS  
0206 31 OE65 2478 BRW STRTMULTI_1 ; AND MULTIECHO THE PROMPT OUT  
OE68 2479 :  
OE68 2480 : SETUP THE CORRECT CURSOR POSITION FOR THE NO PROMPT CASE  
OE68 2481 :  
00FC C5 3A A4 B0 OE68 2482 40$: MOVW TTY$W_RB_CPZORG(R4),UCB$W_TT_CURSOR(R5)
```

```
OE6E 2484 :  
OE6E 2485 : second part of prompt echoing if necessary.  
OE6E 2486 :  
OE6E 2487 PRMECHO1:  
38 A4 B5 OE6E 2488 TSTW TTY$W_RB_CPZCUR(R4) : SHOULD WE UPDATE THIS CURSOR  
OB 14 OE71 2489 BGTR 50$ : NO IF NOT ZERO  
38 A4 B6 OE73 2490 INCW TTY$W_RB_CPZCUR(R4) : WAS THIS -1?  
OB 19 OE76 2491 BLSS 50$ : NO THEN DON'T RESET THE CURSOR  
38 A4 DDFC C5 B0 OE78 2492 MOVW UCB$W TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4) : RESET CURSOR POSITION  
OE7E 2493 50$: CLR STATE <SKIPCR LF,SKIPLF> : CLEAN THE SKIP STATE UP  
OB A4 95 OE84 2494 TSTB TTY$B_RB_ECHLEN(R4) : DO WE HAVE ANYTHING TO ECHO  
OB 14 OE87 2495 BGTR 70$ : YES THEN ECHO THE SPECIFIED STRING  
OB 19 OE89 2496 BLSS 60$ : LESS THAN 0 EXIT  
FF89 31 OE8B 2497 BRW CLRREST : ELSE THEN CLEAR THE REST OF THE LINE  
OB A4 94 OE8E 2498 60$: CLRB TTY$B_RB_ECHLEN(R4) : CLEAN OUT THE OFFSET JUST IN CASE  
FF33 31 OE91 2499 BRW EXITING  
FF6D 31 OE94 2500 70$: BRW CLRECHO
```



```
OE97 2502 : read verify echoing (echo's a string replacing fill characters with clear characte
OE97 2503 :
OE97 2504 :
53 0B A4 97 OE97 2505 RVECHO: DECB TTY$B_RB_ECHLEN(R4) : TAKE THIS CHARACTER OUT
14 1A 19 OE9A 2506 BLSS 208 : NO MORE THEN MOVE THE CURSOR
64 14 B4 9A OE9C 2507 MOVZBL @TTY$B_RB_ECHSTR(R4),R3 : GET A CHARACTER
14 14 A4 D1 OEA0 2508 CMPL TTY$B_RB_ECHSTR(R4),TTY$B_RB_TXT(R4); IS THIS PAST THE PROMPT
53 0A 19 OEA4 2509 BLSS 108 : NO THEN DON'T CHANGE CHARACTERS
53 47 A4 91 OEA6 2510 CMPB TTY$B_RB_RVFFIL(R4),R3 : IS THIS A FILL CHARACTER
04 12 OEAA 2511 BNEQ 108 : NO THEN CONTINUE
53 46 A4 9A OEAC 2512 MOVZBL TTY$B_RB_RVFCLR(R4),R3 : OTHERWISE ECHO THE CLEAR CHARACTER
14 A4 D6 OEB0 2513 108: INCL TTY$B_RB_ECHSTR(R4) : UPDATE THE COUNT
F2D1 31 OEB3 2514 208: BRW FORMAT CHAR : ELSE FORMAT THE CHARACTER
FF06 31 OEB6 2515 BRW MOVIFNEC : MOVE IF NECESSARY
OEB9 2516
```

```

      0EB9 2518 :
      0EB9 2519 : : NON-ANSI CRT THEN SIMULATE CLEAR TO END OF LINE
      0EB9 2520 :
      0EB9 2521 : : SIMCEOL:
      0EB9 2522 : BBC
      0EB8 2523 : #TT$V_SCOPE, - : CLEAR TO END OF LINE IS MEANINGLESS
      0EBE 2524 : UCBSL_DEVDEPEND(R5),MOVECURSOR; ON HARD COPY TERMINALS
      0EC5 2525 : UCBSW_TT_CURSOR(R5),UCBSW_DEVBUFSIZ(R5),R3
      0EC8 2526 : R3,#8 : OUTPUT IN GROUPS OF 8
      0ECA 2527 : 10$ :
      0ED1 2528 : TTY$A_TAB,R4 : SETUP TO ECHO 8
      0ED4 2529 : STRTMULTI
      0ED4 2530 10$: MOVZBW #TTY$K_ER_MOVECURSOR,TTY$W_RB_MODE(R4); NEXT MOVE THE CURSOR
      0ED8 2531 : #1,R3,R1 : SUBTRACT 1 FROM THE COUNT
      0EDC 2532 : TTY$A_TAB+1,R4
      0EE3 2533 : BRW STRTMULTI_1 : AND ECHO IT

53 42 A5 8C 44 A5 0C E1 0EB9 2518 :
      00FC C5 A3 0EB8 2523 :
      08 53 B1 0EC5 2525 :
      0A 15 0EC8 2526 :
54 00000000'EF 9E 0ECA 2527 :
      018C 31 0ED1 2528 :
      44 A4 05 98 0ED4 2529 :
      51 53 01 C3 0ED8 2531 :
54 00000001'EF 9E 0EDC 2532 :
      0188 31 0EE3 2533 :
```

TTYCHARO
V04-000

J 10
- Terminal driver character output routi 16-SEP-1984 02:21:31 VAX/VMS Macro V04-00 Page 80
EDITREAD - READ EDITING STATE 5-SEP-1984 04:16:19 [TTDRVR.SRC]TTYCHARO.MAR;1 (66)

```

      OEE6 2535 :
      OEE6 2536 : set the current cursor position
      OEE6 2537 :
      OEE6 2538 UPDCURSOR:
00FC C5 38 A4 80 OEE6 2539      MOVW  TTY$W_RB_CPZCUR(R4),UCBSW TT CURSOR(R5); UPDATE THE CURSOR POSITION
      FED8 31 OEE6 2540      BRW    EXITING - THEN EXIT

```

```
0EEF 2542 .SBTTL Read service routines
0EEF 2543 :++
0EEF 2544 BACK_TAB
0EEF 2545
0EEF 2546 Calculates the cursor postion of the beginning of a tab while
0EEF 2547 at the end of the tab.
0EEF 2548
0EEF 2549 Implicit inputs:
0EEF 2550
0EEF 2551 R5 UCB address
0EEF 2552 TTY$W_RB_LINOFF the current location off of LIN
0EEF 2553 TTY$S_RB_LIN the address of the beginning of this line
0EEF 2554 TTY$S_RB_TXT the address of the beginning of the data buffer
0EEF 2555 TTY$A_RB_PRM offset from svapte to the beginning of the prompt
0EEF 2556 TTY$W_RB_CPZORG cursor postion when the read started
0EEF 2557
0EEF 2558 Outputs:
0EEF 2559 R3 Mod 8 of the cursor position
0EEF 2560 :--
0EEF 2561 BACK_TAB:
54 78 A5 DD 0EEF 2562 PUSHL R4 : SAVE R4
53 30 A4 3C 0EF1 2563 MOVL UCBSL_SVAPTE(R5),R4 : GET THE ADDRESS OF THE READ BUFFER
53 2C B443 9E 0EF5 2564 MOVZWL TTY$W_RB_LINOFF(R4),R3 : GET THE CURRENT LOCATION
7E D4 0EFE 2565 MOVAB @TTY$C_RB_LIN(R4)(R3),R3 : POINT TO END OF DATA
64 2C A4 D1 0F00 2566 CLRL -(SP) : SET UP CURSOR COUNTER
06 13 0F04 2567 CMPL TTY$S_RB_LIN(R4),TTY$S_RB_TXT(R4) : ARE WE ON THE FIRST LINE?
54 2C A4 D0 0F06 2568 BEQL 10$ : YES THEN CONTINUE
04 11 0F0A 2569 MOVL TTY$S_RB_LIN(R4),R4 : ELSE JUST USE THE BEGINNING OF THIS ONE
54 4A A4 9E 0F0C 2570 BRB 20$ : AND CONTINUE ON
54 54 53 D1 0F10 2571 10$: MOVAB TTY$S_RB_DATA(R4),R4 : POINT TO DATA START
23 13 0F13 2572 20$: CMPL R3,R4 : BUFFER START?
0D 73 91 0F15 2573 BEQL 40$ : THEN REFERENCE POINT FOUND
2C 13 0F18 2574 CMPB -(R3),#TTY$C_CR : CARRIAGE RETURN?
09 63 91 0F1A 2575 BEQL 50$ : IF EQL THEN REFERENCE POINT FOUND
27 13 0F1D 2576 CMPB (R3),#TTY$C_TAB : TAB?
1B 63 91 0F1F 2577 BEQL 50$ : IF EQL THEN REFERENCE POINT FOUND
0F 13 0F22 2578 CMPB (R3),#TTY$C_ESCAPE : IS THIS AN ESCAPE
9B 8F 63 91 0F24 2579 BEQL 30$ : YES THEN REMOVE THE ESCAPE SEQUENCE
09 13 0F28 2580 CMPB (R3),#TTY$C_CSI : CSI'S COUNT ALSO
20 63 91 0F2A 2581 BEQL 30$
E1 1F 0F2D 2582 CMPB (R3),#TTY$C_BLANK : CURSOR CHANGE CHARACTER?
6E D6 0F2F 2583 BLSSU 20$ : IF LSSU THEN NO
DD 11 0F31 2584 INCL (SP) : ADJUST FAKE CURSOR
0F33 2585 BRB 20$ : CONTINUE
0F33 2586 :
0F33 2587 : ESCAPE INTRODUCER FOUND
0F33 2588 :
005C 30 0F33 2589 30$: BSBW ESCAPE_TAB : REMOVE THE ESCAPE SEQUENCE
D8 11 0F36 2590 BRB 20$ : THEN CONTINUE
0F38 2591 :
0F38 2592 : RAN OUT OF BUFFER WITHOUT FINDING A REFERENCE POINT
0F38 2593 :
53 78 A5 D0 0F38 2594 40$: MOVL UCBSL_SVAPTE(R5),R3 : GET THE ADDRESS OF THE READ BUFFER
2C A3 63 D1 0F3C 2595 CMPL TTY$S_RB_TXT(R3),TTY$S_RB_LIN(R3) : THIS IS NOT THE FIRST LINE
04 12 0F40 2596 BNEQ 50$ : USE ZERO BASED ORIGIN.
6E 3A A3 A0 0F42 2597 ADDW TTY$W_RB_CPZORG(R3),(SP) : ELSE USE ORITONAL ORIGIN
0F46 2598 :
```

TTYCHAR0
V04-000

L 10

- Terminal driver character output routi 16-SEP-1984 02:21:31 VAX/VMS Macro V04-00 Page 82
Read service routines 5-SEP-1984 04:16:19 [TTDRVR.SRC]TTYCHAR0.MAR;1 (67)

53	BE	FFFFFFF8	BF	CB	OF46	2599	:	FOUND THE REFERENCE POINT THEN CONTINUE
					OF46	2600	:	
					OF46	2601	:	
					OF4E	2602	:	
					OF51	2603	:	
					OF52	2604	:	

54	BED0	05	50s:	BICL3	#^XOFFFFFFFF8,(SP)+,R3	:	GET MOD 8 OF CURSOR
				POPL	R4	:	RESTORE R4
				RSB			


```
OF52 2606 :++
OF52 2607 :CHAR_SIZE
OF52 2608 :
OF52 2609 :DESCRIPTION:
OF52 2610 :
OF52 2611 :    GIVEN A CHARACTER IN R3 THIS ROUTINE WILL DETERMINE THE NUMBER
OF52 2612 :    OF CURSOR POSITIONS THE CHARACTER WILL TAKE UP.
OF52 2613 :
OF52 2614 :INPUTS:
OF52 2615 :    R1 = 0 FOR FOWARD CHARACTER SIZE -1 FOR BACKWARD CHARACTER SIZE
OF52 2616 :    R3 = CHARACTER
OF52 2617 :    R5 = ADDRESS OF THE UCB
OF52 2618 :
OF52 2619 :OUTPUTS:
OF52 2620 :
OF52 2621 :    R3 = THE LENGTH IN CURSOR POSITIONS OF THIS CHARACTER
OF52 2622 :    R5 = UCB ADDRESS
OF52 2623 :
OF52 2624 :--
OF52 2625 :
OF52 2626 :CHAR_SIZE:
OF52 2627 :CASE W^TTY$A_TYPE[R3],LIMIT=#1@TTY$V_CH_SPEC,TYPE=B,<-
OF52 2628 :    10$,- : Back space
OF52 2629 :    20$,- : tab
OF52 2630 :    5$,- : linefeed (no action)
OF52 2631 :    5$,- : Vtab (no action)
OF52 2632 :    5$,- : form (no action)
OF52 2633 :    60$> : <CR>
OF52 2634 :BITB #<TTY$M_CH_CTRL!TTY$M_CH_SPEC!TTY$M_CH_CTRL2!TTY$M_CH_CTRL3>,-
0000'CF43 04 12 OF68 2635 : W^TTY$A_TYPE[R3] : TEST FOR SPECIAL
53 01 9A OF6C 2636 : BNEQ 5$ : NON SPACING CHARACTER
OF52 2637 : MOVZBL #1,R3 : OTHERWISE ONLY 1 CHARACTER POSITION
OF52 2638 : RSB
OF52 2639 :
OF52 2640 5$: CLRL R3 : CLEAR R3 OUT
OF52 2641 RSB : AND RETURN
OF52 2642 :
OF52 2643 10$: MNEGL #1,R3 : BACKSPACE THEN BACKUP
OF52 2644 RSB
OF52 2645 :
OF52 2646 20$: : TAB
53 00FC C5 03 00 EF OF79 2647 : EXTZV #0,#3,UCB$W_TT_CURSOR(R5),R3: GET HORIZONTAL POINTER
OF52 2648 : TSTL R1 : IS THIS A BACKWARD CALCULATION
OF52 2649 : BGEQ 23$ : NO THEN SUBTRACT
OF52 2650 : BSBW BACK TAB : CALCULATE THE MOD 8 OF THE CURSOR
53 08 FF68 30 OF84 2651 : SUBL3 R3,#8,R3 : NORMALIZE THE RESULT
OF52 2652 : RSB : AND RETURN
OF52 2653 :
OF52 2654 60$: MOVZWL UCB$W_TT_CURSOR(R5),R3 : CARRIAGE RETURN SHOULD ZERO CURSOR
OF52 2655 : RSB : POSITION
```

```
0F92 2657 :++  
0F92 2658 : ESCAPE_TAB  
0F92 2659 :  
0F92 2660 :     Handle the removal of escape sequences from a prompt  
0F92 2661 :     when deleting tabs.  
0F92 2662 :  
0F92 2663 :     Implicit inputs:  
0F92 2664 :  
0F92 2665 :         R3 the address of the escape character  
0F92 2666 :         R5 THE UCB ADDRESS  
0F92 2667 :         4(SP) the internal count  
0F92 2668 :  
0F92 2669 :     Implicit outputs:  
0F92 2670 :  
0F92 2671 :         4(SP) updated internal count  
0F92 2672 :  
0F92 2673 :     All registers are saved  
0F92 2674 :--  
0F92 2675 : ESCAPE_TAB:  
51 08 AE DD 0F92 2676 : PUSHL R1 : SAVE A SCRATCH REGISTER  
18 BB 0F94 2677 : MOVL 8(SP),R1 : RESTORE THE INTERNAL COUNT  
54 53 DO 0F98 2678 : PUSHR #*M<R3,R4> : SAVE THE REST OF THE REGISTERS  
53 84 9A 0F9A 2679 : MOVL R3,R4 : MOVE THE CURRENT POSITION TO R4  
51 DD 0F9D 2680 : MOVZBL (R4)+,R3 : GET THE CHARACTER THAT INTRODUCES THE SEQUENCE  
F4F7 30 0FA0 2681 : PUSHL R1 : SAVE R1  
0103 C5 51 90 0FA2 2682 : BSBW ESCINIT : INIT THE ESCAPE SEQUENCE RULES  
51 8ED0 0FA5 2683 : MOVB R1,UCB$B_TT_ESC(R5) :  
51 0FAD 2684 : POPL R1 : RESTORE R1  
51 D7 0FAD 2685 :  
53 84 9A 0FAF 2686 20$: DECL R1 : DECREMENT THE COUNT  
F4F8 30 0FB2 2687 : MOVZBL (R4)+,R3 : GET THE CHARACTER AND MOVE TO THE NEXT  
F6 14 0FB5 2688 : BSBW ESCSYNTAX : CHECK THE SYNTAX  
18 BA 0FB7 2689 : BGTR 20$ : CONTINUE IF NECESSARY  
08 AE 51 DO 0FB9 2690 : POPR #*M<R3,R4> : SEQUENCE COMPLETE THEN RESTORE THE REGISTERS  
51 8ED0 0FBD 2691 : MOVL R1,8(SP) : PUT THE COUNT BACK  
51 0FC0 2692 : POPL R1 : RESTORE THE LAST REGISTER  
05 0FC0 2693 : RSB : AND NOW RETURN
```

```
OFC1 2695 :++
OFC1 2696 : FIND_BOL - FIND THE BEGINNING OF THIS LINE
OFC1 2697 :
OFC1 2698 : Description:
OFC1 2699 :
OFC1 2700 :     Given a string this routine will find the offset to the character
OFC1 2701 :     that will end up in the first character position of the bottom line
OFC1 2702 :     of the screen
OFC1 2703 :
OFC1 2704 :
OFC1 2705 : IMPLICIT INPUTS:
OFC1 2706 :
OFC1 2707 :     R2 = ADDRESS OF THE UNIT STATE VECTOR
OFC1 2708 :     R4 = ADDRESS OF THE READ BUFFER
OFC1 2709 :     R5 = ADDRESS OF THE UCB
OFC1 2710 :     TTY$$_RB_TXT
OFC1 2711 :     TTY$$_RB_LIN
OFC1 2712 :     TTY$$_RB_PRMLEN
OFC1 2713 :     TTY$$_RB_TXTOFF assumed non-zero
OFC1 2714 :     TTY$$_RB_PRM
OFC1 2715 :     TTY$$_RB_LINOFF
OFC1 2716 :     TTY$$_RB_LINREST assumed zero
OFC1 2717 :
OFC1 2718 : IMPLICIT OUTPUTS:
OFC1 2719 :
OFC1 2720 :     TTY$$_RB_LIN address of the first character in this line of data
OFC1 2721 :     TTY$$_RB_LINOFF offset from LIN to the end of the line
OFC1 2722 :     R3 is destroyed.
OFC1 2723 :
OFC1 2724 : --
OFC1 2725 : FIND_BOL:
OFC1 2726 :     CLRW    UCB$$_TT_CURSOR(R5)      ; CLEAN THE CURSOR POSITION
OFC1 2727 :     FIND_BOL_NOCLEAR:
OFC1 2728 :     CLRL    R1                        ; CLEAN R1
OFC1 2729 :     MOVL    TTY$$_RB_TXT(R4),TTYS$$_RB_LIN(R4); START WITH THE BEGINNING
OFC1 2730 :     PUSH    #M<R6,R7>                ; GET A FEW SCRATCH REGISTERS
OFC1 2731 :     CLRL    R6                        ; CLEAR R6, THE BEGINNING OFFSET
OFC1 2732 :     ADDW    TTY$$_RB_PRMLEN(R4),TTYS$$_RB_TXTOFF(R4),R7; GET THE ENDING OFFSET
OFC1 2733 :     BBCC    #TTY$$_R5_WRAP,TTYS$$_RB_RDSTATE(R4),210$; CLEAR THE WRAP STATE
OFC1 2734 :     210$:  CMPW    R6,R7                ; ARE WE DONE?
OFC1 2735 :     BGEQ    240$                      ; YES THEN EXIT
OFC1 2736 :     MOVZBL  TTY$$_RB_PRM(R4)[R6],R3    ; GET THE CHARACTER
OFC1 2737 :     INCL    R6                        ; MOVE TO THE NEXT CHARACTER
OFC1 2738 :     CMPB    #TTY$$_ET_ESCAPE,TTY$$_CCLIST[R3]
OFC1 2739 :     BEQL    250$                      ; HANDLE ESCAPE SEQUENCES
OFC1 2740 :     CMPB    #TTY$$_CR,R3              ; A CR ZEROS CURSOR
OFC1 2741 :     BEQL    230$                      ; SO DO SO
OFC1 2742 :     BSBW    CHAR_SIZE                 ; OTHER WISE GET THE CURSOR POSITION
OFC1 2743 :     ADDW    R3,UCB$$_TT_CURSOR(R5)    ; ADD IN THE POSITION
OFC1 2744 :     CMPW    UCB$$_TT_CURSOR(R5),UCB$$_DEVBUFSIZ(R5); IS THIS A WRAP
OFC1 2745 :     BLSS    210$                      ; NO THEN GO ON
OFC1 2746 :     SUBW    UCB$$_DEVBUFSIZ(R5),UCB$$_TT_CURSOR(R5); SUBTRACT OUT THE
OFC1 2747 :     LINE LENGTH
OFC1 2748 :     MOVAB   TTY$$_RB_PRM(R4)[R6],R3    ; GET THE ADDRESS OF THE WRAP
OFC1 2749 :     220$:  BBSS    #TTY$$_R5_WRAP,TTYS$$_RB_RDSTATE(R4),225$; SET UP THE WRAP CHARACTER
OFC1 2750 :     225$:  CMPL    R3,TTY$$_RB_TXT(R4) ; ARE WE PAST THE PROMPT?
OFC1 2751 :     BLSS    210$                      ; YES THEN CONTINUE
OFC1 2752 :     MOVL    R3,TTY$$_RB_LIN(R4)      ; AND THIS IS THE LINE
```

00FC C5 B4 OFC1 2725 CLRW UCB\$\$_TT_CURSOR(R5) ; CLEAN THE CURSOR POSITION

2C A4 51 D4 OFC5 2726 FIND_BOL_NOCLEAR:

00C0 8F D0 OFC7 2727 CLRL R1 ; CLEAN R1

57 3C A4 34 A4 A1 OFD1 2728 MOVL TTY\$\$_RB_TXT(R4),TTYS\$\$_RB_LIN(R4); START WITH THE BEGINNING

00 2A A4 00 E5 OFD7 2729 PUSH #M<R6,R7> ; GET A FEW SCRATCH REGISTERS

57 56 D4 OFCF 2730 CLRL R6 ; CLEAR R6, THE BEGINNING OFFSET

53 4A A446 9A OFE1 2731 ADDW TTY\$\$_RB_PRMLEN(R4),TTYS\$\$_RB_TXTOFF(R4),R7; GET THE ENDING OFFSET

00000000'EF43 04 91 OFE8 2732 BBCC #TTY\$\$_R5_WRAP,TTYS\$\$_RB_RDSTATE(R4),210\$; CLEAR THE WRAP STATE

53 0D 91 OFF2 2733 210\$: CMPW R6,R7 ; ARE WE DONE?

53 0D 91 OFF2 2734 BGEQ 240\$; YES THEN EXIT

00FC C5 53 A0 OFFA 2735 MOVZBL TTY\$\$_RB_PRM(R4)[R6],R3 ; GET THE CHARACTER

42 A5 00FC C5 B1 OFFF 2736 INCL R6 ; MOVE TO THE NEXT CHARACTER

00FC C5 42 A5 A2 1007 2737 CMPB #TTY\$\$_ET_ESCAPE,TTY\$\$_CCLIST[R3]

53 4A A446 9E 100D 2738 BEQL 250\$; HANDLE ESCAPE SEQUENCES

00 2A A4 00 E2 1012 2739 CMPB #TTY\$\$_CR,R3 ; A CR ZEROS CURSOR

64 53 D1 1017 2740 BEQL 230\$; SO DO SO

2C A4 53 D0 101C 2741 BSBW CHAR_SIZE ; OTHER WISE GET THE CURSOR POSITION

00FC C5 53 A0 OFFA 2742 ADDW R3,UCB\$\$_TT_CURSOR(R5) ; ADD IN THE POSITION

42 A5 00FC C5 B1 OFFF 2743 CMPW UCB\$\$_TT_CURSOR(R5),UCB\$\$_DEVBUFSIZ(R5); IS THIS A WRAP

00FC C5 42 A5 A2 1007 2744 BLSS 210\$; NO THEN GO ON

53 4A A446 9E 100D 2745 SUBW UCB\$\$_DEVBUFSIZ(R5),UCB\$\$_TT_CURSOR(R5); SUBTRACT OUT THE

00 2A A4 00 E2 1012 2746 LINE LENGTH

64 53 D1 1017 2747 MOVAB TTY\$\$_RB_PRM(R4)[R6],R3 ; GET THE ADDRESS OF THE WRAP

2C A4 53 D0 101C 2748 220\$: BBSS #TTY\$\$_R5_WRAP,TTYS\$\$_RB_RDSTATE(R4),225\$; SET UP THE WRAP CHARACTER

00FC C5 53 A0 OFFA 2749 225\$: CMPL R3,TTY\$\$_RB_TXT(R4) ; ARE WE PAST THE PROMPT?

42 A5 00FC C5 B1 OFFF 2750 BLSS 210\$; YES THEN CONTINUE

00FC C5 42 A5 A2 1007 2751 MOVL R3,TTY\$\$_RB_LIN(R4) ; AND THIS IS THE LINE

```
BA 11 1020 2752 BRB 210$ ; THEN CONTINUE
1021 2753
53 4A A446 9E 1022 2754 230$: MOVAB TTY$A_RB_PRM(R4)[R6],R3 ; GET THE ADDRESS OF THE WRAP
00FC C5 B4 1023 2755 CLRW UCB$W_TT_CURSOR(R5) ; CLEAR CURSOR ON A <CR>
E5 11 1024 2756 BRB 220$ ; NOW CONTINUE
1025 2757
1026 2758 ;
1027 2759 ; ESCAPE SEQUENCE INTRODUCER
1028 2760
51 DD 1029 2761 250$: PUSHL R1 ; SAVE R1
F46A 30 102F 2762 BSBW ESCINIT ; INIT THE SYNTAX RULES
0103 C5 51 90 1032 2763 MOVAB R1,UCB$B_TT_ESC(R5) ;
51 BED0 1037 2764 POPL R1 ; RESTORE THE REGISTER
103A 2765 260$:
57 56 B1 103A 2766 CMPW R6,R7 ; ARE WE FINISHED
OF 18 103D 2767 BGEQ 240$ ; YES THEN EXIT
53 4A A446 90 103F 2768 MOVAB TTY$A_RB_PRM(R4)[R6],R3 ; GET THE NXT CHARACTER
56 D6 1044 2769 INCL R6 ; MOVE TO THE NEXT CHARACTER
F464 30 1046 2770 BSBW ESCSYNTAX ; AND CHECK THE SYNTAX
EF 14 1049 2771 BGTR 260$ ; MORE THEN CONTINUE
FF8E 31 104B 2772 BRW 210$ ; NO MORE THEN GO BACK
104E 2773
53 4A A446 9E 104E 2774 240$: MOVAB TTY$A_RB_PRM(R4)[R6],R3 ; GET THE ADDRESS OF THIS CHARACTER
53 2C A4 C2 1053 2775 SUBL TTY$A_RB_LIN(R4),R3 ; GET THE OFFSET TO THE LAST CHARACTER
30 A4 53 B0 1057 2776 MOVW R3,TTY$W_RB_LINOFF(R4) ; AND LEAVE IT there
00C0 8F BA 105B 2777 POPR #^M<R6,R7> ; RESTORE THE REGISTERS
05 105F 2778 RSB
```



```
1060 2780 :++
1060 2781 : STRTMULTI - START A MULTIECHO STRING GIVEN THE ADDRESS OF AN ASCIC STRING
1060 2782 :
1060 2783 : FUNCTIONAL DESCRIPTION:
1060 2784 :
1060 2785 : THIS ROUTINE STARTS THE SPECIFIED MULTIECHO STRING ON A UNIT.
1060 2786 : THE COUNT OF THE STRING IS PLACED IN UCB$W TT_MULTILEN, AND THE
1060 2787 : ADDRESS OF THE FIRST CHARACTER IN THE STRING IS PLACED IN UCB$L TT_MULTI.
1060 2788 : THEN STATE MULTI IS SET AND MULTIECHO IS CALLED.
1060 2789 :
1060 2790 : STRTMULTI_1 - starts a multiecho string given the address and a length
1060 2791 : of a string (address in R4, length in R1)
1060 2792 :
1060 2793 : INPUTS
1060 2794 :
1060 2795 : R2 = ADDRESS OF THE UNIT STATE VECTOR
1060 2796 : R4 = ADDRESS OF A COUNTED MULTIECHO STRING (ASCIC)
1060 2797 : R5 = UCB ADDRESS
1060 2798 :
1060 2799 : OUTPUTS:
1060 2800 :
1060 2801 : R2 = ADDRESS OF THE UNIT STATE VECTOR
1060 2802 : R3 = FIRST CHARACTER
1060 2803 : R5 = UCB ADDRESS
1060 2804 : R4 DESTROYED
1060 2805 : --
1060 2806 : STRTMULTI::
1060 2807 : MOVW UCB$W TT_MULTILEN(R5),UCB$W TT_SMLTLEN(R5); SAVE LENGTH
1060 2808 : MOVZBW (R4)+,UCB$W TT_MULTILEN(R5) ;GET THE LENGTH INTO ALTLINA
1060 2809 : BRB LOADADR
1060 2810 : STRTMULTI_1::
1060 2811 : MOVW UCB$W TT_MULTILEN(R5),UCB$W TT_SMLTLEN(R5); SAVE LENGTH
1060 2812 : MOVW R1,UCB$W TT_MULTILEN(R5) ; MOVE IN THE NEW LENGTH
1060 2813 : LOADADR:
1060 2814 : MOVL UCB$L TT_MULTI(R5),UCB$L TT_SMLT(R5); SAVE THE ADDRESS
1060 2815 : MOVL R4,UCB$L TT_MULTI(R5) ;THEN THE ADDRESS INTO MULTI
1060 2816 : SET_STATE <MULTI> ;NEXT SET MULTIECHO STATE
1060 2817 : brw multiechoing
```

00DE C5 00DC C5 B0 1060 2807
00DC C5 84 9B 1067 2808
0C 11 106C 2809
00DE C5 00DC C5 B0 106E 2810
00DC C5 51 B0 1075 2812
00E0 C5 00D8 C5 D0 107A 2813
00D8 C5 54 D0 1081 2815
F3D1 3i 108A 2817


```
108D 2819 :++
108D 2820 : TABRIGHT - CHECK FOR A TAB TO THE RIGHT OF THE CURSOR POSITION
108D 2821 :
108D 2822 : DESCRIPTION:
108D 2823 :
108D 2824 :     This routine will detect a tab character to the right of the cursor
108D 2825 :     position and set a flag indicating that insert mode is illegal. This
108D 2826 :     prevents having to recalculate tab and wrap position for each character.
108D 2827 :
108D 2828 : Inputs:
108D 2829 :     R4 - Read packet address
108D 2830 :
108D 2831 : Outputs:
108D 2832 :
108D 2833 :     TABRIGHT state bit set if a tab is to the right of the cursor
108D 2834 :     reset if not.
108D 2835 :     All registers preserved.
108D 2836 : --
108D 2837 : TABRIGHT:
108D 2838 :     PUSHR    #^M<R0,R1>          ; SAVE SOME WORKING REGISTERS
108F 2839 :     CLR STATE TABRIGHT          ; CLEAR THE STATE
1094 2840 :     MOVZWL   TTY$W_RB_LINOFF(R4),R1 ; GET THE OFFSET INTO THE STRING
1098 2841 :     LOCC     #TTY$C_TAB,TTY$W_RB_LINREST(R4),@TTY$L_RB_LIN(R4)[R1];
109F 2842 :                                     ; TRY TO FIND A TAB
109F 2843 :     BEQL     10$                ; DID WE FIND A TAB?
10A1 2844 :     SET STATE TABRIGHT          ; YES THEN SET THE STATE
10A6 2845 :     POPR     #^M<R0,R1>          ; RESTORE THE REGISTERS
10A8 2846 :     RSB                                     ; AND RETURN
10A9 2847 :
```

2C B441 51 30 A4 3C 1094 2840
 32 A4 09 3A 1098 2841
 05 13 109F 2842
 03 BA 10A1 2843
 05 10A6 2844
 10A8 2845
 10A9 2846

```
10A9 2849 :++
10A9 2850 : UPDATE_CURSOR
10A9 2851 :
10A9 2852 : DESCRIPTION:
10A9 2853 :
10A9 2854 : WILL SETUP THE NECESSARY SEQUENCE OF EVENTS TO HAVE
10A9 2855 : THE CURSOR MOVED TO THE PLACE SPECIFIED BY TTY$W_RB_CPZCUR
10A9 2856 :
10A9 2857 : INPUTS:
10A9 2858 :
10A9 2859 : R2 = ADDRESS OF THE UNIT STATE VECTOR
10A9 2860 : R5 = UCB ADDRESS
10A9 2861 : TTY$W_RB_CPZCUR = THE FINAL CURSOR POSITION
10A9 2862 : UCB$W_TT_CURSOR = THE CURRENT POSITION.
10A9 2863 :
10A9 2864 : OUTPUT:
10A9 2865 :
10A9 2866 : R2 = ADDRESS OF THE UNIT STATE VECTOR
10A9 2867 : R4 = ADDRESS OF THE READ PACKET
10A9 2868 : R5 = UCB ADDRESS
10A9 2869 : R3 DESTROYED
10A9 2870 :--
10A9 2871 : UPDATE_CURSOR:
10A9 2872 : MOVL UCB$L_SVAPTE(R5),R4 ; GET THE READ PACKET ADDRESS
10AD 2873 : SUBW3 TTY$W_RB_CPZCUR(R4),- ; SUBTRACT THE FINAL CURSOR POSITION
10B0 2874 : UCB$W_TT_CURSOR(R5),- ; FROM THE CURRENT CURSOR POSITION
10B3 2875 : UCB$W_TT_BSLEN(R5) ; AND ECHO THAT MANY BACKSPACES
10B6 2876 : CMPW UCB$W_TT_CURSOR(R5),-
10BA 2877 : UCB$W_DEVBUFSIZ(R5) ; ARE WE AT THE END OF THE LINE
10BC 2878 : 5$ ; NO THEN CONTINUE
10BE 2879 : DECW UCB$W_TT_CURSOR(R5) ; KEEP THE POSITION CORRECT
10C2 2880 : DECW UCB$W_TT_BSLEN(R5) ; ELSE THE TERMINAL DOES A BACKSPACE
10C6 2881 : BEQL 40$ ; NO BACKSPACES THEN EXIT
10C8 2882 : 5$:
10C8 2883 : BBC #TT2$V_ANSICRT,UCB$L_DEVDEPND2(R5),30$; NOT AN ANSI TERMINAL THEN BR
10CD 2884 : CMPW #8,UCB$W_TT_BSLEN(R5) ; CAN WE OPTIMIZE ON AN ANSI TERMINAL
10D2 2885 : BGTR 30$ ; NO THEN OUTPUT BACKSPACES
10D4 2886 : PUSHR #^M<R0,R1,R2,R3,R4,R5> ; SAVE THE REGISTERS
10D6 2887 : MOVZBL TTY$A_ANSIBACKUP,R1 ; GET THE LENGTH OF THE STRING
10DD 2888 : INCL R1 ; ADD IN THE COUNT TOO.
10DF 2889 : MOVCL R1,TTY$A_ANSIBACKUP,TTY$Q_RB_ECHOAREA(R4); MOVE THE CHARACTER
10E8 2890 : STRING IN TO PLACE
10E8 2891 : POPR #^M<R0,R1,R2,R3,R4,R5> ; RESTORE THE REGISTERS
10EA 2892 : CLRL -(SP) ; GET ENOUGH ROOM FOR THE DIVIDEND
10EC 2893 : MOVZWL UCB$W_TT_BSLEN(R5),-(SP) ; AND PUT IT THE DIVISOR
10F1 2894 : MOVAB TTY$Q_RB_ECHOAREA+6(R4),R1 ; GET THE ADDRESS OF THE FIRST BYTE
10F5 2895 : EDIV #10, (SP), (SP), R3 ; GET THE FIRST DIGIT
10FA 2896 : ADDB R3, -(R1) ; PUT IT IN PLACE
10FD 2897 : TSTL (SP) ; DO WE HAVE ANY MORE TO DO
10FF 2898 : BNEQ 10$ ; YES THEN DO SO
1101 2899 : TSTL (SP)+ ; CLEAN THE STACK
1103 2900 : TSTL (SP)+
1105 2901 : MOVW TTY$W_RB_CPZCUR(R4),UCB$W_TT_CURSOR(R5); UPDATE THE CURSOR POSITION
1108 2902 : MOVZBL #TTY$K_ER_UPDCURSOR,TTY$W_RB_MODE(R4); SETUP THE EDITREAD STATE
110B 2903 : SET STATE EDITREAD ; And setup the edit read state
110B 2904 : MOVAB TTY$Q_RB_ECHOAREA(R4),R4 ; GET THE ECHO AREA ADDRESS
110F 2905 : BRW STRMOLTI ; and start the multiechoing

54 78 A5 D0 10A9 2872
38 A4 A3 10AD 2873
00FC C5 10B0 2874
0100 C5 10B3 2875
00FC C5 B1 10B6 2876
42 A5 10BA 2877
0A 1F 10BC 2878
00FC C5 B7 10BE 2879
0100 C5 B7 10C2 2880
4D 13 10C6 2881
10C8 2882
45 48 A5 18 E1 10C8 2883
0100 C5 08 B1 10CD 2884
3E 14 10D2 2885
3F BB 10D4 2886
51 00000000'EF 9A 10D6 2887
OC A4 00000000'EF 51 D6 10DD 2888
51 28 10DF 2889
3F BA 10E8 2890
7E 0100 C5 10EA 2892
51 12 A4 9E 10F1 2894
53 6E 6E 0A 7B 10F5 2895
71 53 80 10FA 2896
6E D5 10FD 2897
F4 12 10FF 2898
8E D5 1101 2899
8E D5 1103 2900
00FC C5 38 A4 B0 1105 2901
1108 2902
110B 2903
54 OC A4 9E 110B 2904
FF4E 31 110F 2905
```

TTYCHARO
V04-000

- Terminal driver character output rout1 16-SEP-1984 02:21:31 VAX/VMS Macro V04-00 Page 90
Read service routines 5-SEP-1984 04:16:19 [TTDRVR.SRC]TTYCHARO.MAR;1 (73)

EEED 31 1112 2906
1112 2907 308: SET_STATE <BACKSPACE>
1115 2908 408: BRW- GETNEXTCHAR ; AND START MULTIECHOING BACKSPACES
; GET THE NEXT CHARACTER

```
1118 2910 :++
1118 2911 :ZERO_CPZORG - HANDLE CASE WHERE WE NEED A FRESH LINE
1118 2912 :
1118 2913 :DESCRIPTION:
1118 2914 :
1118 2915 :   This routine will check and make sure that the original cursor
1118 2916 :   position becomes zero. This will then callback the routine that called it
1118 2917 :   and re-run the operation.
1118 2918 :
1118 2919 :Inputs:
1118 2920 :
1118 2921 :   R4 - READ BUFFER ADDRESS
1118 2922 :   R5 - UCB ADDRESS
1118 2923 :   (SP) - ADDRESS OF A ROUTINE TO CONTINUE WITH
1118 2924 :--
1118 2925 ZERO_CPZORG:
06 2A A4 00 E0 1118 2926 BBS #TTY$V_RS_WRAP,TTY$W_RB_RDSTATE(R4),50$; IF WE WRAPPED THEN
1118 2927 : WE MUST FIX UP THE IO
2C A4 64 D1 111D 2928 CMPL TTY$L_RB_TXT(R4),TTY$L_RB_LIN(R4); DID WE WRAP?
1118 2929 BEQL 60$; NO THEN NO SPECAIL ACTION
14 A4 8E D0 1123 2930 50$: MOVL (SP)+,TTY$L_RB_ECHSTR(R4); SETUP A CALLBACK
F79E 31 1127 2931 BRW CTRLR; AND REFRESH THE IO
05 112A 2932 60$: RSB
```

```
112B 2934 .sbtll Read passall
112B 2935 :++
112B 2936 :
112B 2937 PASS ALL AND NO FILTER HANDELER
112B 2938 :
112B 2939 INPUTS:
112B 2940 R2 = ADDRESS OF THE UNIT STATE VECTOR
112B 2941 R3 = CHARACTER TO INSERT
112B 2942 R4 = READ PACKET ADDRESS
112B 2943 R5 = UCB ADDRESS
112B 2944 :
112B 2945 :--
112B 2946 PASSALL::
112B 2947
51 3C A4 3C 112B 2948 MOVZWL TTY$W_RB_TXTOFF(R4),R1 ; GET THE PLACE TO PUT THIS CHARACTER
00 B441 53 90 112F 2949 MOVB R3,@TTY$C_RB_TXT(R4)[R1]; PUT THE CHARACTER AWAY
1134 2950 IF STATE ESC,1800$ ; ESCAPE THE EXIT
1138 2951 IF NOT STATE PASALL,1900$ ; NOFORMAT THEN UPPER CASING MAY BE NECESSAR
113C 2952 1005$: IF STATE ESCAPE,1045$ ; IF ESCAPE MODE THEN EXIT
62 1C B4 53 E0 1140 2953 1040$: BBS R3,@TTY$L_RB_TERM(R4),1500$; IS THIS A TERMINATOR
40 A4 3C A4 B6 1145 2954 INCW TTY$W_RB_TXTOFF(R4) ; INCREMENT THE COUNT
40 A4 3C A4 B1 1148 2955 1010$: CMPW TTY$W_RB_TXTOFF(R4),TTY$W_RB_TXTSIZ(R4); ARE WE FINISHED
46 1E 114D 2956 BGEQU 1600$ ; YES THEN FINISH
114F 2957
114F 2958 1030$: IF STATE <ESC>,1000$ ; IS THIS THIS ESCAPE
1153 2959 IF NOT STATE PRE,1025$ ; IF PRETYPEAHEAD THEN RETURN
05 1157 2960 1020$: RSB
1158 2961 1025$: IF NOT STATE NOECHO,1027$
10 48 A5 E1 115C 2962 BBC #TTY$V_LOCALECHO,- ; LOCAL ECHO?
06 E0 115E 2963 UCB$L_DEVDEPND2(R5),1026$; NO THEN MUST BE NOECHO
0B 20 A4 1161 2964 BBS #TRMSV_TM_NOECHO,- ; NOECHO SPECIFIED
F07E 31 1163 2965 TTY$L_RB_MOD(R4),1026$ ; ON THE READ
F01B 31 1166 2966 BRW FORMAT_LOCAL ; ELSE LOCAL ECHO IT
EE91 31 1169 2967 1027$: BRW FORMAT_CHAR ; FORMAT THE CHARACTER
0065 31 116C 2968 1000$: IF NOT STATE PRE,1026$
1170 2969 RSB
1171 2970 1026$: BRW GETNEXTCHAR
1174 2971
1174 2972 1045$: BRW 1400$
1177 2973 ; CONVERT LOWER TO UPPER IF NOFORMAT
1177 2974
1177 2975 1900$: MOVL UCB$L_IRP(R5),R1 ; GET THE IRP ADDRESS
BC 20 A1 08 E1 117B 2976 BBC #IOSV_CVTLOW,[RPSW_FUNC(R1),1005$; BR IF CONVERT LOWER TO UPPER
B5 0000 CF43 03 E1 1180 2977 BBC #TTY$V_CH_LOWER,W^TTY$A_TYPE[R3],1005$; BR IF NOT LOWER CHARACTER
53 20 AA 1187 2978 BICW #^X020,R3 ; CONVERT TO UPPER CASE
51 3C A4 3C 118A 2979 MOVZWL TTY$W_RB_TXTOFF(R4),R1 ; GET THE PLACE TO PUT THIS CHARACTER
00 B441 53 90 118E 2980 MOVB R3,@TTY$C_RB_TXT(R4)[R1]; PUT THE CHARACTER AWAY
A7 11 1193 2981 BRB 1005$
1195 2982
1195 2983 1600$: MOVL UCB$L_IRP(R5),R1 ; GET THE IRP ADDRESS
38 A1 B4 1199 2984 CLRW IRP$L_MEDIA(R1) ; SET NO TERMINATOR
119C 2985 SET STATE EOL ; SIGNIFY END OF LINE
B2 04 A2 1A E4 11A0 2986 BBSC #TTY$V_ST_PRE,4(R2),1020$; CLEAR PRE IF NECESSARY AND EXIT
AB 11 11A5 2987 BRB 1030$
11A7 2988
11A7 2989
30 A4 3C A4 B0 11A7 2990 1500$: MOVW TTY$W_RB_TXTOFF(R4),TTY$W_RB_LINOFF(R4); SETUP LINOFF
```



```
66 11 11AC 2991 SET_STATE EOL ; SIGNIFY END OF LINE
11B0 2992 BRB 1440$
11B2 2993
11B2 2994 : ESCAPE IN PROGRESS
11B2 2995
51 58 A5 D0 11B2 2996 1800$: MOVL UCBSL_IRP(R5),R1 ; GET THE IRP ADDRESS
3A A1 B6 11B6 2997 INCW IRPSL_MEDIA+2(R1) ; ADD 1 TO THE SIZE
3C A4 B6 11B9 2998 INCW TTY$W_RB_TXTOFF(R4) ; UP THE OFFSET BECAUSE WE PUT 1 CHAR IN THE
F2EE 30 11BC 2999 BSBW ESCSYNTAX ; CHECK THE SYNTAX OF THIS CHAR
18 14 11BF 3000 BGTR 1830$ ; OK THEN CONTINUE ON
04 13 11C1 3001 BEQL 1810$ ; VALID SEQUENCE CONTINUE
11C3 3002
11C3 3003 SET_STATE BADESC ; A BAD ESCAPE SEQUENCE THEN SAY SO
11C7 3004 1810$: CLR_STATE ESC ; CLEAR THE ESCAPE STATE
3C A4 3A A1 A2 11CC 3005 SUBW IRPSL_MEDIA+2(R1),TTY$W_RB_TXTOFF(R4); SUBTRACT OUT THE SEQUENCE
11D1 3006 SET_STATE <EOC,NOECHO> ; LENGTH AND SET END OF LINE
11D9 3007
0047 31 11D9 3008 1830$: BRW 1450$ ; THEN EXIT
11DC 3009
04 00000000'EF43 91 11DC 3010 1400$: CMPB TTY$A_CCLIST[R3],#TTY$K_ET_ESCAPE; CHECK FOR ESCAPE INTR
5C 12 11E4 3011 BNEQ 1470$ ; ESCAPE SEQUENCE INTRODUCER
53 7D 8F 91 11E6 3012 CMPB #TTY$C_LOWESC1,R3 ; IS THIS AN ALTMODE
06 13 11EA 3013 BEQL 1410$ ; YES THEN HANDLE SPECAILLY
53 7E 8F 91 11EC 3014 CMPB #TTY$C_LOWESC2,R3 ; LOWER ALSO
11 12 11F0 3015 BNEQ 1420$ ; NO THEN MUST BE AN ESC OR CSI
4B 44 A5 07 E0 11F2 3016 1410$: BBS #TTY$V_LOWER,UCBSL_DEVDEPEND(R5),1470$; LOWER CASE TERMINAL
51 3C A4 3C 11F7 3017 MOVZWL TTY$W_RB_TXTOFF(R4),R1 ; GET THE LOCATION OF THIS CHARACTER
00 B441 1B 90 11FB 3018 MOVW #TTY$C_ESCAPE,@TTY$S_RB_TXT(R4)[R1]; AND FORCE THE ESCAPE
53 1B 9A 1200 3019 MOVZBL #TTY$C_ESCAPE,R3 ; AND PUT THE ESCAPE IN R3
3C A4 B6 1203 3020 1420$: INCW TTY$W_RB_TXTOFF(R4) ; ESCAPE INTRO THEN SAY SO.
51 DD 1206 3021 PUSHL R1 ; SAVE
F291 30 1208 3022 BSBW ESCINIT
0103 C5 51 90 120B 3023 MOVW R1,UCBSB_TT_ESC(R5)
51 8ED0 1210 3024 POPL R1
1213 3025 SET_STATE ESC ; SETUP ESCAPE STATE
51 58 A5 D0 1218 3026 1440$: MOVL UCBSL_IRP(R5),R1 ; GET THE IRP ADDRESS
38 A1 53 B0 121C 3027 MOVW R3,IRPSL_MEDIA(R1) ; SETUP THE TERMINATING CHARACTER
3A A1 B6 1220 3028 INCW IRPSL_MEDIA+2(R1) ; AND THE LENGTH
40 A4 3C A4 B1 1223 3029 1450$: CMPW TTY$W_RB_TXTOFF(R4),TTY$W_RB_TXTSIZ(R4); ARE WE FINISHED
15 1E 1228 3030 BGEQU 1460$ ; YES THEN FINISH
122A 3031 IF NOT_STATE EOL,1480$ ; IF NOT END OF LINE THEN RETURN
12 04 A2 1A E4 122E 3032 BBSC #TTY$V_ST_PRE,4(R2),1490$
1233 3033 IF STATE NOECHO,1480$
1237 3034 BBS #TRMSV_TM_TRMNOECHO,-
OC 20 A4 1239 3035 TTY$S_RB_MOD(R4),1495$ ; IF NOT ECHOING TERMINATER THEN EXIT
FF10 31 123C 3036 1480$: BRW 1030$
FF53 31 123F 3037 1460$: BRW 1600$
FEFB 31 1242 3038 1470$: BRW 1040$
FFOF 31 1245 3039 1490$: BRW 1020$
FF26 31 1248 3040 1495$: BRW 1026$
```

```
1248 3042 .SBTTL READ WITH VERIFICATION
1248 3043
1248 3044 ++
1248 3045 RDVERIFY
1248 3046
1248 3047 Description:
1248 3048 Read verify allows programs that wish to do character validation
1248 3049 to issue one IO rather than a QIO per character as was previously the case.
1248 3050
1248 3051 Inputs:
1248 3052
1248 3053 R2 - Unit state vector
1248 3054 R5 - address of the ucb
1248 3055
1248 3056 Implicit inputs:
1248 3057
1248 3058 UCBSL_SVAPTE - The address of the read buffer.
1248 3059 TTYSL_RB_TXT - The address of the first character of the initial string
1248 3060 TTYSW_RB_TXTOFF - Offset to the last character in the initial string
1248 3061 TTYSW_RB_TXTSIZ - Length of the data buffer.
1248 3062 TTYSW_RB_LINOFF - Offset to the end of the field, initial offset.
1248 3063 TTYSL_RB_PIC - The address of the picture string.
1248 3064
1248 3065 RDVERIFY::
1248 3066 MOVL UCBSL_TT_TYPAMD(R5),R4 ; ADDRESS TYPEAHEAD BUFFER
1248 3067 BEQL 40$ ; NO BUFFER THEN THERE IS A FORK
1248 3068 ; PROCESS WAITING SO LET IT GO.
1248 3069 TSTW TTYSW_TA_INAHD(R4) ; ANY TYPEAHEAD DATA?
1248 3070 BNEQ 50$ ; yes then take it out and continue
1248 3071
1248 3072 TYPEAHEAD BUFFER EXHAUSTED
1248 3073
1248 3074 IF_NOT_STATE <TYPFUL>,35$ ; BR IF TYPEAHEAD IS NOT FULL
1248 3075
1248 3076 20$: BRW XON ; CONTINUE
1248 3077 35$:
1248 3078
1248 3079 THE TYPEAHEAD BUFFER HAS NOW BEEN EMPTIED INTO THE SYSTEM BUFFER.
1248 3080 IF PROCESSING A READ WITH ZERO SECOND TIMEOUT, RETURN THE DATA
1248 3081 TO THE USER IMMEDIATELY.
1248 3082
1248 3083 BBC #UCBSV TT TIMO,UCBSW DEVSTS(R5),40$; BR IF NOT READ WITH TIMEOUT.
1248 3084 CMPL TTYSA_MAXTIME,UCBSL_TT_RDUE(R5); ZERO SECOND TIMEOUT?
1248 3085 BNEQ 40$ ; BR IF NOT.
1248 3086 MOVW #SS$ TIMEOUT,UCBSW_BOFF(R5); SET TIMEOUT COMPLETION STATUS.
1248 3087 BSBW TTY$READONE ; COMPLETE REQUEST.
1248 3088
1248 3089 DISMISS INTERRUPT
1248 3090
1248 3091 40$: CLRB UCBSB_TT_OUTYPE(R5) ; SET NO RETURN CHARACTER.
1248 3092 RSB ; and dismiss the interrupt
1248 3093
1248 3094 ACTUALLY REMOVE CHARACTER FROM BUFFER
1248 3095
1248 3096 50$: MOVZBL @TTYSL_TA_GET(R4),R3 ; GET THE CHARACTER
1248 3097 BBCC #TTY$V-ST-ECHAES,4(R2),55$; IS THIS THE FIRST CHARACTER
1248 3098 MOVL UCBSL_SVAPTE(R5),R4 ; GET THE READ PACKET ADDRESS
1248 3099 MOVZBW #TTY$R_ER_AESECHO,TTYSW_RB_MODE(R4); YES THEN ECHO THE AES STRING
```

			128D	3099	SET_STATE EDITREAD	: NOW DO EDITING
FAB7	31		1291	3100	BRW EDITREAD	: AND ECHO THE STRING
			1294	3101	55\$:	
OC A4	B7		1294	3102	DECW TTYSW_TA_INAHD(R4)	: ADJUST NUMBER OF TYPEAHEAD CHARACTERS
			1297	3103		
			1297	3104	...	CHECK FOR BUFFER WRAP AROUND
			1297	3105	...	
06 04 A4	10 A4	F2	1297	3106	AOBLSS TTYSL_TA_END(R4),TTYSL_TA_GET(R4),60\$:	POINTER PAST END?
04 A4	0118 C4	9E	129D	3107	MOVAB TTYSL_TA_DATA(R4),TTYSL_TA_GET(R4);	RESET POINTER
			12A3	3108		
	50	DD	12A3	3109	60\$: PUSH R0	: charo can not destroy R0
54	78 A5	D0	12A5	3110	MOVL UCBSL_SVAPTE(R5),R4	: GET READ BUFFER
			12A9	3111	IF STATE ESC,T0\$: IF ESCAPE SEQUENCE IN PROGRESS THEN HANDLE
20 A4	11	E1	12AD	3112	BBC #TRMSV_TM_R_JUST,TTYSL_RB_MOD(R4),-	BRANCH IF LEFT JUSTIFY MODE
	06		12B1	3113		
	015C	31	12B2	3114	BRW LEFT_JUSTIFY	
	029C	31	12B5	3115	10\$: BRW RIGHT_JUSTIFY	
					BRW ESCAPE_TERM	

```

12B8 3117 : THIS ROUTINE HANDLES LEFT JUSTIFIED FIELDS
12B8 3118 :
12B8 3119 :
12B8 3120 :
12B8 3121 LEFT_JUSTIFY:
12B8 3122 :
12B8 3123     MOVZWL  TTY$W_RB_LINOFF(R4),R1           ; GET CURRENT OFFSET INTO FIELD
12BC 3124 :
12BC 3125     TERMINATE IF ALREADY FIELD FULL - THIS CAN OCCUR ON A NON AUTOTAB
12BC 3126     FIELD WHERE WE DEFER "FULL FIELD" COMPLETION UNTIL ONE MORE
12BC 3127     CHARACTER IS TYPED. CHAR TYPED IS TERMINATOR.
12BC 3128 :
12BC 3129     CMPW    R1,TTY$W_RB_TXTOFF(R4)           ; FIELD FULL?
12C0 3130     BLSS   5$,                               ; IF LSS, NOT YET
12C2 3131     BRW    TERM                             ; YES - INPUT CHAR IS TERM
12C5 3132 5$:
12C5 3133 :
12C5 3134 ; Check to see if this character is valid in this position
12C5 3135 :
12C5 3136     MOVZBL  @TTY$L_RB_PIC(R4)[R1],R0       ; ADDRESS CORRESPONDING PIC VALUE
12CA 3137     BITB    R0,VERIFY_ARRAY[R3]           ; CHAR VALID ?
12D2 3138     BNEQ   10$,                           ; YES
12D4 3139     BRW    TERM                             ; ELSE TERMINATOR
12D7 3140 10$:
12D7 3141 :
12D7 3142 :
12D7 3143     UPCASE IF REQUESTED.
12D7 3144 :
12D7 3145     BBC     #TRMSV_TM_CVTLOW,TTY$L_RB_MOD(R4),13$; NO CONVERT THEN DON'T
12DC 3146     BBC     #TTY$V_CH_LOWER,W@TTY$A_TYPE[R3],13$; is this lower case?
12E3 3147     BICB   #^X20,R3                     ; yes MAKE LOWERCASE UPPER
12E6 3148 13$:
12E6 3149     MOVB    R3,@TTY$L_RB_TXT(R4)[R1]       ; STORE INTO READ BUFFER
12EB 3150 20$:
12EB 3151     INCW    R1                               ; POINT TO NEXT POSITION
12ED 3152     CMPW    R1,TTY$W_RB_TXTOFF(R4)       ; FIELD FULL?
12F1 3153     BGEQ   FULL_1                         ; YES, WITH SINGLE CHARACTER
12F3 3154 :
12F3 3155 ; Check for trailing marker characters to be skipped
12F3 3156 :
12F3 3157     TSTB    @TTY$L_RB_PIC(R4)[R1]           ; NEXT POSITION MARKER?
12F7 3158     BNEQ   OUTPUT_1                       ; NO, OUTPUT 1 CHARACTER
12F9 3159 :
12F9 3160     DECL    R1                               ; POINT TO PREVIOUS POSITION
12FB 3161     MOVAB   @TTY$L_RB_TXT(R4)[R1],-          ; SAVE ADDRESS OF STRING TO OUTPUT
1301 3162     TTY$L_RB_ECHSTR(R4)
1301 3163     MOVZBL  #1,R0                             ; INIT COUNT OF CHARACTERS IN STRING
1304 3164 30$:
1304 3165     INCL    R1                               ; POINT TO NEXT POSITION
1306 3166     CMPW    R1,TTY$W_RB_TXTOFF(R4)       ; FIELD FULL
130A 3167     BLSS   35$,                               ; NO
130C 3168     BRW    FULL_1                             ; YES, SO DON'T OUTPUT
130F 3169     TRAILING MARKERS
130F 3170 35$:
130F 3171     TSTB    @TTY$L_RB_PIC(R4)[R1]           ; NEXT SLOT A MARKER
1313 3172     BNEQ   45$,                               ; NO
1315 3173     INCW    R0                               ; COUNT MARKERS
1317 3174     BRB    30$                             ; CONTINUE LOOKING

```



```
0B A4 50 90 1319 3174 45$:
1319 3175      MOVW  R0,TTY$B_RB_ECHLEN(R4)      : SAVE STRING LENGTH
131D 3176      BRW   OUTPUT_S                    : OUTPUT STRING
131D 3177
131D 3178 OUTPUT_S:                             : OUTPUT A STRING
131D 3179      MOVW  R1,TTY$W_RB_LINOFF(R4)        : SAVE NEW OFFSET
1321 3180      IF STATE NOECHO,DONE                : NOECHO THEN DON'T ECHO
1325 3181      CLRW  TTY$W_RB_LINREST(R4)          : DON'T CHANGE CURSOR POSITION
44 A4 32 A4 B4 1325 3181      MOVZBW #TTY$K_ER_RVECHO,TTY$W_RB_MODE(R4); : SETUP TO ECHO
9B 1328 3182
132C 3183      SET_STATE EDITREAD
18 11 1330 3184      BRB   DONE
1332 3185
1332 3186 OUTPUT_1:                             : OUTPUT CHARACTER IN R3
1332 3187      MOVW  R1,TTY$W_RB_LINOFF(R4)        : SAVE NEW OFFSET
47 A4 53 91 1336 3188      CMPB  R3,TTY$B_RB_RVFFIL(R4) : USER TYPE FILLCHAR
04 12 133A 3189      BNEQ  EXIT                    : NO
53 46 A4 90 133C 3190      MOVW  TTY$B_RB_RVFCLR(R4),R3 : YES - REP W/CLR CHAR
1340 3191
1340 3192 EXIT:
50 8ED0 1344 3193      IF STATE NOECHO,DONE                : NO EDCHO THEN DON'T ECHO
EE3D 31 1347 3195      POPL  R0
134A 3196      BRW   FORMAT_CHAR                    : FORMAT THE CHARACTER ELSEWISE
134A 3197
134A 3198 DONE:
50 8ED0 134A 3198      POPL  R0                        : RESTORE R0
ECB0 31 134D 3199      BRW   TTY$GETNEXTCHAR          : AND GET THE NEXT CHARACTER
1350 3200 FULL_1:                             : FIELD FULL, OUTPUT SINGLE CHARACTE
1350 3201
1350 3202      IF NOT AUTOTAB, DONT END NOW - LET HIM HIT A CHAR
1350 3203
1350 3204 BBC   #TRMSV_TM_AUTO_TAB,-
20 A4 DD 1352 3205      TTY$B_RB_MOD(R4),-
1354 3206      OUTPUT_1
1355 3207      : ECHO LAST CHAR
30 A4 51 B0 1355 3208      MOVW  R1,TTY$W_RB_LINOFF(R4)        : SAVE NEW OFFSET
1359 3209      SET_STATE EOL                                : TELL THEM WE ARE DONE
54 58 A5 D0 135D 3210      MOVW  UCB$B_IRP(R5),R4            : GET IRP ADDRESS
38 A4 D4 1361 3211      CLRL  IRP$B_MEDIA(R4)                : SIGNAL NO TERMINATOR
DA 11 1364 3212      BRB   EXIT
1366 3213
1366 3214 TERM:
30 A4 51 B0 1366 3215      MOVW  R1,TTY$W_RB_LINOFF(R4)        : SAVE ENDING OFFSET
136A 3216      IF STATE NOFLTR,3$                        : SKIP EDITING IF NOFLTR
00000000'EF43 03 91 136E 3217      CMPB  #TTY$K_ET_DELEFT,TTY$A_CCLIST[R3]; : IS THIS A DELETE CHARACTER
46 13 1376 3218      BEQL  DELEFT-JUST                    : YES THEN DELETE
1378 3219 3$:
00000000'EF43 04 91 137C 3220      IF NOT_STATE ESCAPE,4$ : ESCAPE PROCESSING THEN
24 13 1384 3221      CMPB  #TTY$K_ET_ESCAPE,TTY$A_CCLIST[R3]; : ESCAPE ?
1386 3222      BEQL  10$                                    : YES
50 58 A5 D0 138A 3223      SET_STATE EOL                    : SET END OF LINE
38 A0 53 9B 138E 3224      MOVW  UCB$B_IRP(R5),R0
3A A0 01 90 1392 3225      MOVZBW R3,IRP$B_MEDIA(R0)          : TERMINATOR
3B A0 51 90 1396 3226      MOVW  #1,IRP$B_MEDIA+2(R0)        : TERM LENGTH
51 3C A4 3C 139A 3227      MOVW  R1,IRP$B_MEDIA+3(R0)        : OFFSET TO INVALID CHAR
139E 3228      MOVZWL TTY$W_RB_TXTOFF(R4),R1                : GET THE OFFSET TO THE END
18 A4 00 B441 9E 139E 3229      MOVAB @TTY$B_RB_TXT(R4)[R1],TTY$B_RB_PIC(R4); : OF THE BUFFER
18 B4 53 90 13A4 3230      MOVW  R3,@TTY$B_RB_PIC(R4)          : SETUP THE END OF THE BUFFER
: STORE IN BUFFER
```



```
AO 11 13A8 3231 BRB DONE
13AA 3232
13AA 3233 : SIGNAL ESCAPE SEQUENCE ACTIVE
13AA 3234
13AA 3235 10$:
13AA 3236 SET STATE ESC
51 DD 13AF 3237 PUSH R1 : SAVE R1
FOE8 30 13B1 3238 BSBW ESCINIT : INIT THE ESCAPE SEQUENCE RULES
0103 C5 51 90 13B4 3239 MOV R1,UCB$B_TT_ESC(R5)
51 8ED0 13B9 3240 POPL R1 : RESTORE R1
CC 11 13BC 3241 BRB 5$ : SKIP SETTING END
13BC 3242
13BE 3243 : DELETE ONE CHARACTER TO THE LEFT
13BE 3244
13BE 3245 : R1 = offset into the line
13BE 3246
13BE 3247
13BE 3248 DELEFT_JUST:
50 D4 13BE 3249 CLRL R0 : A COUNTER
51 D7 13C0 3250 DECL R1 : move left 1 character
OC 19 13C2 3251 BLSS 100$ : NO CHARACTER TO DELETE THEN DON'T
50 D6 13C4 3252 10$: INCL R0 : MOVE BACK 1 CHARACTER
18 B441 95 13C6 3253 TSTB @TTY$B_RB_PIC(R4)[R1] : IS THIS A MARKER?
07 12 13CA 3254 BNEQ 20$ : NO THEN SHIFT
51 D7 13CC 3255 DECL R1 : ELSE MOVE OVER IT
F4 18 13CE 3256 BGEQ 10$ : NO NON-MARKER THEN IGNORE THE DEL
FF77 31 13D0 3257 100$: BRW DONE : ...
13D3 3258
3C A4 30 A4 B1 13D3 3259 20$: CMPW TTY$B_RB_LINOFF(R4),TTY$B_RB_TXTOFF(R4); WERE WE ABOUT TO TERMINATE
03 12 13D8 3260 BNEQ 25$ : NO THEN PROCESS NORMALLY
50 01 D0 13DA 3261 MOVL #1,R0 : ELSE ONLY GO BACK 1 PLACE
13DD 3262 : DUE TO SPECAIL EOF CONDITIONS
30 A4 51 B0 13DD 3263 25$: MOVW R1,TTY$B_RB_LINOFF(R4) : UPDATE THE OFFSET
13E1 3264 IF STATE NOECHO,50$
38 A4 00FC C5 50 A3 13E5 3265 SUBW3 R0,UCB$B_TT_CURSOR(R5),TTY$B_RB_CPZCUR(R4); SET THE FINAL CURSOR
0100 C5 50 B0 13EC 3266 MOVW R0,UCB$B_TT_BSLEN(R5) : AND TELL HOW MANY CHARACTERS TO
13F1 3267 : BACKSPACE OVER
13F1 3268 SET STATE <BACKSPACE,EDITREAD> : THEN OUTPUT OTHER STUFF AND RETURN
14 A4 00 B441 9E 13F6 3269 MOVAB @TTY$B_RB_TXT(R4)[R1],TTY$B_RB_ECHSTR(R4); GET THE ADDRESS OF WHERE
08 A4 01 90 13FC 3270 MOV B #1,TTY$B_RB_ECHLEN(R4) : THE LENGTH OF THE STRING TO OUTPUT
44 A4 0A 9B 1400 3271 MOVZBW #TTY$B_ER_RVECHO,TTY$B_RB_MODE(R4); SET THE MODE OF THE EDITING
32 A4 01 D0 1404 3272 MOVL #1,TTY$B_RB_LINREST(R4) : TELL THE MOVE TO RESTORE THE POSIT
1408 3273
00 B441 47 A4 90 1408 3274 50$: MOV B TTY$B_RB_RVFFIL(R4),@TTY$B_RB_TXT(R4)[R1]; AND FILL THE BLANK
140E 3275 : WITH THE FILL CHARACTER
FF39 31 140E 3276 BRW DONE
```

```
1411 3278 RIGHT_JUSTIFY:
1411 3279
51 30 A4 32 1411 3280 CVTWL TTY$W_RB_LINOFF(R4),R1 ; GET CURRENT OFFSET
03 18 1415 3281 BGEQ 3$ ; IF WE ARE BEOND THE
00BD 31 1417 3282 ; BOUNDARY THEN TERMINATE
141A 3283
50 18 B441 9A 141A 3284 3$: BRW RIGHT_TERM
0D 12 141F 3285 MOVZBL @TTY$L_RB_PIC(R4)[R1],R0 ; GET PICTURE VALUE
51 D7 1421 3286 BNEQ 5$ ; NOT A MARKER
F5 18 1423 3287 DECL R1 ; CHECK FOR END
51 D4 1425 3288 BGEQ 3$ ; NOT YET
30 A4 51 B0 1427 3289 CLRL R1
0076 31 142B 3290 MOVW R1,TTY$W_RB_LINOFF(R4) ; SAVE CURRENT OFFSET
00000000'EF43 50 93 142E 3292 5$: BRW FIELD_FULL ; FULL OF MARKERS
03 12 1436 3293 BITB R0,VERIFY_ARRAY[R3] ; IS THIS CHARACTER VALID HERE
009C 31 1438 3294 BNEQ 10$ ; OK
143B 3295
143B 3296 ; SHIFT DATA IN INPUT FIELD
143B 3297
143B 3298
143B 3299 10$:
30 A4 51 B0 143B 3300 MOVW R1,TTY$W_RB_LINOFF(R4) ; SAVE CURRENT OFFSET
51 D4 143F 3301 CLRL R1 ; INIT DESTINATION POINTER
1441 3302
1441 3303 ; FIND FIRST VALID DESTINATION
1441 3304
1441 3305
1441 3306 20$:
18 B441 95 1441 3307 TSTB @TTY$L_RB_PIC(R4)[R1] ; MARKER?
0B 12 1445 3308 BNEQ 30$ ; NO, POSITION IS OK
51 D6 1447 3309 INCL R1 ; LOOK FOR NEW DESTINATION
3C A4 51 B1 1449 3310 CMPW R1,TTY$W_RB_TXTOFF(R4) ; PAST END?
F2 1F 144D 3311 BLSSU 20$ ; NO, IT IS STILL OK
0052 31 144F 3312 BRW FIELD_FULL
1452 3313
1452 3314 30$:
1452 3315 ; IF NONCLR CHAR IN LEFTMOST DATA POSITION, FIELD IS ALREADY FULL, MUST
1452 3316 ; BE NON-AUTOTAB. TERMINATE WITH OVERFLOWING CHAR
1452 3317
1452 3318
1452 3319 CMPB @TTY$L_RB_TXT(R4)[R1],TTY$B_RB_RVFFIL(R4);THIS A CLR CHAR?
3C A4 13 1458 3319 BEQL 35$ ; YES - NOT FULL YET
30 A4 B0 145A 3320 MOVW TTY$W_RB_TXTOFF(R4),- ; SHOW OVERFLOW OFFSET
76 11 145D 3321 TTY$W_RB_LINOFF(R4) ; IF FIELD FULL
51 DD 1461 3322 BRB RIGHT_TERM ; AND QUIT
50 51 D0 1461 3324 35$: PUSHL R1 ;SAVE OFFSET TO LEFT BYTE
1463 3325
1466 3326 MOVL R1,R0 ; INIT SOURCE POINTER
1466 3327
1466 3328 ; FIND NEW SOURCE
1466 3329
1466 3330 40$:
30 A4 50 D6 1466 3331 INCL R0
50 50 B1 1468 3332 CMPW R0,TTY$W_RB_LINOFF(R4) ; PAST END?
12 1A 146C 3333 BGTRU 100$ ; YES THEN SHIFT COMPLETE
18 B440 95 146E 3334 TSTB @TTY$L_RB_PIC(R4)[R0] ; MARKER?
```

```
00 B441 00 B440 90 1472 3335 BEQL 40$ ; YES, LOOK AGAIN
51 50 D0 1474 3336 MOVB @TTY$LB_TXT(R4)[R0],@TTY$LB_TXT(R4)[R1]; SHIFT THE DATA
E6 11 147B 3337 MOVL R0,R1 ; OLD SOURCE IS NEW DESTINATION
1480 3338 BRB 40$
1480 3339 ; SHIFT COMPLETE
1480 3340
1480 3341
1480 3342 100$:
1480 3343
1480 3344
1480 3345
1480 3346 UPCASE IF REQUESTED.
0A 20 08 E1 1480 3347 BBC #TRMSV TM CVTLOW,- ; SKIP IF UPCASE NOT
03 0000'CF43 03 E1 1482 3347 TTY$LB_MOD(R4),105$ REQUESTED
53 20 8A 1485 3348 BBC #TTY$V CH_LOWER,W^TTY$A_TYPE[R3],105$; is this lower case?
148C 3349 BICB #^X20,R3 ; yes MAKE LOWERCASE UPPER
148F 3350 105$:
148F 3351
148F 3352 MOVB R3,@TTY$LB_TXT(R4)[R1] ; STORE NEW CHARACTER INTO
1494 3353 ; END POSITION
1494 3354 POPL R1 ; GET START POSITION
46 A4 00 B441 91 1497 3355 CMPB @TTY$LB_TXT(R4)[R1],TTY$B_RB_RVFCLR(R4); START WITH CLR CHAR?
10 13 149D 3356 BEQL RIGHT_ECHO ; YES - FLD NOT FULL
12 E1 149F 3357 BBC #TRMSV TM AUTO TAB,- ; FLD FULL - NONAUTOTAB
OB 20 A4 14A1 3358 TTY$LB_MOD(R4),RIGHT_ECHO ; JUST ECHO
14A4 3359 ; FLD FULL - AUTOTAB
14A4 3360 ; FALL THROUGH
14A4 3361
14A4 3362
14A4 3363 FIELD_FULL:
14A4 3364 SET_STATE EOL
14A8 3365 10$:
51 58 A5 D0 14A8 3366 MOVL UCB$LB_IRP(R5),R1 ; GET IRP ADDRESS
38 A1 D4 14AC 3367 CLRL IRP$LB_MEDIA(R1) ; SIGNAL NO TERMINATOR
14AF 3368
14AF 3369 RIGHT_ECHO:
00FC C5 B4 14AF 3370 CLRW UCB$W TT CURSOR(R5)
14B3 3371 IF STATE NOECHO,GODONE ; NO ECHO THEN DON'T ECHO
50 8ED0 14B7 3372 POPL R0
32 A4 B4 14BA 3373 CLRW TTY$W RB LINREST(R4)
44 A4 0A 9B 14BD 3374 MOVZBW #TTY$K ER RVECHO,TTY$W_RB_MODE(R4); ECHO THE LINE
14 A4 4A A4 9E 14C1 3375 MOVAB TTY$A_RB_PRM(R4),TTY$LB_ECHSTR(R4); THE ADDRESS OF THE STRING
30 A4 34 A4 81 14C6 3376 ADDB3 TTY$W_RB_PRLLEN(R4),TTY$W_RB_LINOFF(R4),-; GET THE LENGTH TO ECHO
OB A4 14CB 3377 TTY$B_RB_ECHLEN(R4)
OB A4 96 14CD 3378 INCB TTY$B_RB_ECHLEN(R4) ; ACCOUNT FOR ZERO BASE
14D0 3379 SET_STATE EDITREAD
F874 31 14D4 3380 BRW EDITREAD
14D7 3381
14D7 3382 .ENABLE LSB
14D7 3383
14D7 3384 RIGHT_TERM:
00000000'EF43 03 91 14DB 3385 IF STATE NOFLTR,3$ ; SKIP EDITING IF NOFLTR
4D 13 14E3 3386 CMPB #TTY$K ET DELEFT,TTY$A_CCLIST[R3]; IS THIS A DELETE CHARACTER
14E5 3387 BEQL DELRIGHT JUST ; YES THEN DELETE
00000000'EF43 04 91 14E9 3388 3$: IF NOT_STATE ESCAPE,4$ ; ESCAPE PROCESSING THEN
2B 13 14F1 3389 CMPB #TTY$K ET_ESCAPE,TTY$A_CCLIST[R3]; ESCAPE ?
7E A5 3C A4 B0 14F3 3390 BEQL 10$ ; YES
3391 4$: MOVW TTY$W_RB_TXTOFF(R4),UCB$W_BCNT(R5); FILL REQUIRED FIELD
```

```
14FB 3392 SET_STATE <EOL> ; SET END OF LINE
14FC 3393 5$:
14FC 3394 MOVL UCBSL_IRP(R5),R1 ; GET IRP ADDRESS
3B A1 53 9B 1500 3395 MOVZBW R3,IRPSL_MEDIA(R1) ; TERMINATOR
3A A1 01 90 1504 3396 MOVB #1,IRPSL_MEDIA+2(R1) ; TERM LENGTH
30 A4 90 1508 3397 MOVB TTY$W_RB_LINOFF(R4),- ;
3B A1 150B 3398 IRPSL_MEDIA+3(R1) ; OFFSET TO INVALID CHAR
51 3C A4 3C 150D 3399 MOVZWL TTY$W_RB_TXTOFF(R4),R1 ; GET THE ADDRESS OF THE END
1B A4 00 B441 9E 1511 3400 MOVAB @TTY$C_RB_TXT(R4)[R1],TTY$C_RB_PIC(R4) ; OF THE INITIAL STRING
1B B4 53 90 1517 3401 MOVB R3,@TTY$C_RB_PIC(R4) ; AND STORE THE CHARACTER THERE
FE2C 31 151B 3402 GODONE: BRW DONE ; THEN EXIT
151E 3403
151E 3404 :: SIGNAL ESCAPE SEQUENCE ACTIVE
151E 3405
151E 3406 10$:
151E 3407 SET STATE ESC ; ANNOUNCE ESCAPE STATE
1523 3408 PUSHL R1 ; SAVE R1
EF74 30 1525 3409 BSBW ESCINIT ; INIT THE ESCAPE SEQUENCE RULES
0103 C5 51 90 1528 3410 MOVB R1,UCBSB_TT_ESC(R5) ;
51 8ED0 152D 3411 POPL R1 ; RESTORE R1
CA 11 1530 3412 BRB 5$ ; SKIP SETTING END
1532 3413
1532 3414 .DISABLE LSB
1532 3415
1532 3416 :: DELETE RIGHT JUSTIFIED
1532 3417
1532 3418 DELRIGHT JUST:
1532 3419
50 51 D0 1532 3420 MOVL R1,R0 ; GET THE CURRENT LOCATION
50 D7 1535 3421 10$: DECL R0 ; MOVE RIGHT 1 CHARACTER
12 19 1537 3422 BLSS 20$ ; ECHO THE LINE
18 B440 95 1539 3423 TSTB @TTY$C_RB_PIC(R4)[R0] ; IS THIS A MARKER?
F6 13 153D 3424 BEQL 10$ ; YES THEN SKIP IT
00 B441 D0 B440 90 153F 3425 MOVB @TTY$C_RB_TXT(R4)[R0],- ; MOVE THE CHARACTER
1546 3426 @TTY$C_RB_TXT(R4)[R1]
51 50 D0 1546 3427 MOVL R0,R1
EA 11 1549 3428 BRB 10$
154B 3429
47 A4 90 154B 3430 20$: MOVB TTY$B_RB_RVFFIL(R4),- ; MOVE A FILL CHARACTER FROM THE LEFT
00 B441 154E 3431 @TTY$C_RB_TXT(R4)[R1] ;
FF5B 31 1551 3432 BRW RIGHT_ECHO ; NOW ECHO THE CHARACTER
```



```

1554 3434 .SBTTL READ VERIFY ESCAPE TERMINATOR ROUTINE
1554 3435
1554 3436 ++
1554 3437 DESCRIPTION:
1554 3438 CALLED ON ALL THE CHARACTERS FOLLOWING THE ESCAPE
1554 3439
1554 3440 INPUTS:
1554 3441
1554 3442 R2 - UNIT STATE VECTOR
1554 3443 R4 - ADDRESS OF THE READ BUFFER
1554 3444 R5 - UCB ADDRESS
1554 3445
1554 3446 IMPLICIT INPUTS:
1554 3447
1554 3448 IRPSL_MEDIA+2 - CURRENT OFFSET TO IN THE BUFFER (ACCESSED AS A BYTE)
1554 3449 TTYSL_RB_PIC - THE ADDRESS OF THE ESCAPE CHARACTER IN THE BUFFER
1554 3450 TTYSW_RB_TXTSIZ - LENGTH OF THE USERS BUFFER
1554 3451 --
1554 3452 ESCAPE_TERM:
1554 3453 MOVL UCB$$_IRP(R5),R0 ; GET IRP ADDRESS
1558 3454 MOVZBL IRPSL_MEDIA+2(R0),R1 ; GET CURRENT OFFSET
155C 3455
155C 3456 CMPW R1,TTYSW_RB_TXTSIZ(R4) ; DO WE HAVE ROOM FOR THIS
1560 3457 BGEQU 30$ ; NO THEN TELL THEM ABOUT IT
1562 3458 MOVW R3,TTYSL_RB_PIC(R4)[R1] ; STORE THE DATA
1567 3459 INCB IRPSL_MEDIA+2(R0) ; BUMP TERM LENGTH
156A 3460
156A 3461 BSBW ESCSYNTAX ; PARSE ESCAPE SEQUENCE
156D 3462 BGTR 10$ ; SEQUENCE OK, CONTINUE
156F 3463 BLSS 20$ ; SEQUENCE ERROR
1571 3464
1571 3465 ; SEQUENCE COMPLETED NORMALLY
1571 3466 CLR_STATE ESC ; CLEAR THE ESCAPE STATE THEN
1576 3467
1576 3468 30$: SET_STATE EOL ; SET END OF LINE
157A 3469 10$:
157A 3470 BRW DONE
157D 3471
157D 3472 20$:
157D 3473 SET_STATE BADESC ; ESCAPE SEQ ERROR
1581 3474 BRB 30$ ; TELL THE WORLD

```


TTYCHAR0
V04-000

G 12
- Terminal driver character output routi 16-SEP-1984 02:21:31 VAX/VMS Macro V04-00 Page 103
End of module 5-SEP-1984 04:16:19 [TTDRVR.SRC]TTYCHAR0.MAR;1 (80)

1583 3475 .SBTTL End of module
1583 3476 TT_END:
1583 3477 .END

TTYCHARO
Symbol table

H 12
- Terminal driver character output rout 16-SEP-1984 02:21:31 VAX/VMS Macro V04-00 Page 104
5-SEP-1984 04:16:19 [TTDRVR.SRC]TTYCHARO.MAR;1 (80)

ADJUST_CURSOR	000001A7	R	02
AESECHO	00000DF5	R	02
BACKSPACING	000003F4	R	02
BACK_CHAR	0000088B	R	02
BACK_TAB	00000EEF	R	02
BSPACE	00000204	R	02
BUFEMPTY	0000060B	R	02
BURST	00000099	R	02
CALLBACK	00000DDC	R	02
CARRIAGE	0000020B	R	02
CHAR_SIZE	00000F52	R	02
CHECK_BUFFER	00000593	R	02
CLRECHO	00000E04	R	02
CLRREST	00000E17	R	02
CNT	= 00000001		
COMECHO	00000E0C	R	02
CROUTPUT	0000020F	R	02
CTRLR	000008C8	R	02
CTRLU	00000908	R	02
CTRLZ	0000029C	R	02
CURSOROVRFLOW	00000406	R	02
CVTLOW	0000064D	R	02
DELCHAR	000009C3	R	02
DELEFT_JUST	000013BE	R	02
DELESCAPE	00000B41	R	02
DELETE_WORD	00000B5A	R	02
DELRIGHT_JUST	00001532	R	02
DISMISS	0000065A	R	02
DONE	0000134A	R	02
DROP	00000299	R	02
DROP_CHAR	00000B57	R	02
ECHLINE	00000E44	R	02
EDITINGCHAR	00000803	R	02
EDITREAD	00000D4B	R	02
EOLSEEN	0000043B	R	02
ESCAPE	000002C5	R	02
ESCAPE_CHAR	00000BF2	R	02
ESCAPE_TAB	00000F92	R	02
ESCAPE_TERM	00001554	R	02
ESCINIT	0000049C	RG	02
ESCINPROG	0000065F	R	02
ESCSYNTAX	000004AD	R	02
ESCSYNTAX_D	000004C3	R	02
EXESGL_ABSTIM	*****	X	02
EXIT	00001340	R	02
EXITING	00000DC7	R	02
E_SYNTAX	000004D0	RG	02
F	= 00000000		
FIELD_FULL	000014A4	R	02
FILLING	00000450	R	02
FIND_BOL	00000FC1	R	02
FIND_BOL_NOCLEAR	00000FC5	RG	02
FORM	000003C6	R	02
FORMAT	000000D6	R	02
FORMAT_CHAR	00000187	R	02
FORMAT_LOCAL	000001E7	R	02
FORMAT_X	0000013C	R	02

FORWARD_CHAR	00000C51	R	02
FULL_1	00001350	R	02
GETNEXTCHAR	00000005	R	02
GODEL	00000B54	R	02
GODONE	0000151B	R	02
GODROP	00000C89	R	02
GOOUT	00000201	R	02
INDELST	0000073B	R	02
INSERT_CHAR	00000571	R	02
INTEXT	0000003D	R	02
INTEXT1	00000048	R	02
IOSM_TRMNOECHO	= 00001000		
IOSV_CVTLOW	= 00000008		
IOSV_NOECHO	= 00000006		
IOSV_TRMNOECHO	= 0000000C		
IRPSC_MEDIA	= 00000038		
IRPSW_FUNC	= 00000020		
LEFT_JUSTIFY	000012B8	R	02
LFOUTPUT	0000031A	R	02
LINEFEED	00000315	R	02
LOADADR	0000107A	R	02
MOVECURSOR	00000E4A	R	02
MOVEREADATA	00000526	R	02
MOVE_BOL	00000C8C	R	02
MOVE_EOL	00000CDB	R	02
MOVIFNEC	00000DBF	R	02
MULTIECHOING	0000045E	R	02
NORMAL	00000D6C	R	02
NOTA	00000619	R	02
OPEN	00000752	R	02
OUTPUTANDWAIT	00000148	R	02
OUTPUTANDWAIT1	0000014D	R	02
OUTPUT_1	00001332	R	02
OUTPUT_S	0000131D	R	02
PASSALC	0000112B	RG	02
POWERREST	00000042	R	02
PRMECHO	00000E51	R	02
PRMECHO1	00000E6E	R	02
QUOTE	000007EE	R	02
QUOTING	00000D06	R	02
RDVERIFY	0000124B	RG	02
RECALL	00000D0E	R	02
RECALLING	0000085C	R	02
RESTART	*****	X	02
RIGHT_ECHO	000014AF	R	02
RIGHT_JUSTIFY	00001411	R	02
RIGHT_TERM	000014D7	R	02
RVECHO	00000E97	R	02
SENDLINEFEED	00000493	R	02
SIMCEOL	00000EB9	R	02
SS\$_CONTROLO	= 00000609		
SS\$_DATAOVERUN	= 00000838		
SS\$_NORMAL	= 00000001		
SS\$_TIMEOUT	= 0000022C		
STRMULTI	00001060	RG	02
STRMULTI_1	0000106E	RG	02
STR_EXIT	0000009E	R	02

TTYCHARO
Symbol table

1 12
- Terminal driver character output routi 16-SEP-1984 02:21:31 VAX/VMS Macro V04-00
5-SEP-1984 04:16:19 [TTDRVR.SRC]TTYCHARO.MAR;1

Page 105
(80)

STR_TIMESET	000000AF	R	02
T	= 00000028		
TAB	00000358	R	02
TABRIGHT	00001080	R	02
TAB_BUFEMPTY	00000507	R	02
TAB_CVTLOW	00000520	R	02
TAB_DISMISS	00000504	R	02
TAB_EDITINGCHAR	00000516	R	02
TAB_EOLSEEN	000004FA	R	02
TAB_ESCINPROG	00000519	R	02
TAB_GETNEXT	00000510	R	02
TAB_INDELST	0000050A	R	02
TAB_LOCAL	000003A5	R	02
TAB_NOTA	000004F7	R	02
TAB_OPEN	00000523	R	02
TAB_PASSALL	0000050D	R	02
TAB_QUOTE	00000513	R	02
TERM	00001366	R	02
TERMFOUND	000005BC	R	02
TOGGELINOV	00000D3E	R	02
TRMSV_TM_AUTO_TAB	= 00000012		
TRMSV_TM_CVTLOW	= 00000008		
TRMSV_TM_NOECHO	= 00000006		
TRMSV_TM_NORECALL	= 00000010		
TRMSV_TM_R_JUST	= 00000011		
TRMSV_TM_TRMNOECHO	= 0000000C		
TTSV_COWER	= 00000007		
TTSV_MECHFORM	= 00000013		
TTSV_MECHTAB	= 00000008		
TTSV_SCOPE	= 0000000C		
TTSV_WRAP	= 00000009		
TTS_VT100	= 00000060		
TTS_VTSX	= 00000040		
TT2SV_ANSICRT	= 00000018		
TT2SV_DECCRT	= 0000001D		
TT2SV_EDITING	= 0000000C		
TT2SV_LOCALECHO	= 00000000		
TTYSA_ANSIBACKUP	*****	X	02
TTYSA_ANSICEL	*****	X	02
TTYSA_ANSI_DEOL	*****	X	02
TTYSA_ANSI_UPCEL	*****	X	02
TTYSA_CCLIST	*****	X	02
TTYSA_CTRLR	*****	X	02
TTYSA_CTRLU	*****	X	02
TTYSA_DELCRTTAB	*****	X	02
TTYSA_ESCAPE	*****	X	02
TTYSA_ESCINIT	*****	X	02
TTYSA_EXITECHO	*****	X	02
TTYSA_FCNTKN	*****	X	02
TTYSA_FORM	*****	X	02
TTYSA_LONGFORM	*****	X	02
TTYSA_MAXTIME	*****	X	02
TTYSA_PREFIX	*****	X	02
TTYSA_RB_PRM	= 0000004A		
TTYSA_SPACEBACK	*****	X	02
TTYSA_TAB	*****	X	02
TTYSA_TA_RCL	= 00000018		

TTYSA_TYPE	*****	X	02
TTYSA_VTAB	*****	X	02
TTYSA_WORDTERM	*****	X	02
TTYSB_RB_ECHLEN	= 00000008		
TTYSB_RB_RVFCLE	= 00000046		
TTYSB_RB_RVFFIL	= 00000047		
TTYSC_BLANK	= 00000020		
TTYSC_BS	= 00000008		
TTYSC_CR	= 0000000D		
TTYSC_CSI	= 0000009B		
TTYSC_CTRLB	= 00000002		
TTYSC_CTRLZ	= 0000001A		
TTYSC_DELETE	= 0000007F		
TTYSC_DOLLAR	= 00000024		
TTYSC_ESCAPE	= 0000001B		
TTYSC_LF	= 0000000A		
TTYSC_LOWESC1	= 0000007D		
TTYSC_LOWESC2	= 0000007E		
TTYSC_TAB	= 00000009		
TTYSGETNEXTCHAR	= 00000000	RG	02
TTYSK_EDITNORMAL	= 00000004		
TTYSK_ER_AESECHO	= 00000009		
TTYSK_ER_CLRECHO	= 00000001		
TTYSK_ER_CLRREST	= 00000006		
TTYSK_ER_ECHLINE	= 00000002		
TTYSK_ER_EXITING	= 00000004		
TTYSK_ER_MOVECURSOR	= 00000005		
TTYSK_ER_PRMECHO	= 00000007		
TTYSK_ER_PRMECHO1	= 00000008		
TTYSK_ER_RVECHO	= 0000000A		
TTYSK_ER_SIMCEOL	= 0000000B		
TTYSK_ET_BACK_CHAR	= 00000005		
TTYSK_ET_DELEFT	= 00000003		
TTYSK_ET_ESCAPE	= 00000004		
TTYSK_ET_FORWARD_CHAR	= 00000006		
TTYSK_ET_RECALL	= 0000000B		
TTYSK_ET_TERMINATE	= 0000000E		
TTYSK_ET_UNUSED	= 0000000D		
TTYSK_MAXESCTKN	*****	X	02
TTYSL_RB_AES	= 00000024		
TTYSL_RB_DATA	= 0000004A		
TTYSL_RB_ECHSTR	= 00000014		
TTYSL_RB_LIN	= 0000002C		
TTYSL_RB_MOD	= 00000020		
TTYSL_RB_PIC	= 00000018		
TTYSL_RB_TERM	= 0000001C		
TTYSL_RB_TXT	= 00000000		
TTYSL_TA_DATA	= 00000118		
TTYSL_TA_END	= 00000010		
TTYSL_TA_GET	= 00000004		
TTYSL_WB_END	= 00000020		
TTYSL_WB_IRP	= 00000024		
TTYSL_WB_NEXT	= 0000001C		
TTYSM_CH_CTRL	= 00000020		
TTYSM_CH_CTRL2	= 00000080		
TTYSM_CH_CTRL3	= 00000040		
TTYSM_CH_SPEC	= 00000010		

TTYCHARO
Symbol table

J 12
- Terminal driver character output routi 16-SEP-1984 02:21:31 VAX/VMS Macro V04-00
5-SEP-1984 04:16:19 [TTDRVR.SRC]TTYCHARO.MAR;1

Page 106
(80)

TTY\$M-ST-BACKSPACE = 00000020
TTY\$M-ST-BADESC = 00000100
TTY\$M-ST-CTRLO = 00000001
TTY\$M-ST-CTRLR = 00040000
TTY\$M-ST-CURSOR = 00000008
TTY\$M-ST-DEL = 00000002
TTY\$M-ST-EDITING = 00100000
TTY\$M-ST-EDITREAD = 00000200
TTY\$M-ST-EOL = 00000100
TTY\$M-ST-ESC = 00000080
TTY\$M-ST-ESCAPE = 00000800
TTY\$M-ST-ESC O = 00004000
TTY\$M-ST-FILC = 00000004
TTY\$M-ST-MULTI = 00000040
TTY\$M-ST-NINTMULTI = 08000000
TTY\$M-ST-NL = 00000200
TTY\$M-ST-NOECHO = 00000008
TTY\$M-ST-NOFLTR = 00000040
TTY\$M-ST-OVERSTRIKE = 00800000
TTY\$M-ST-OVRFLO = 00010000
TTY\$M-ST-PASALL = 00000004
TTY\$M-ST-PRE = 04000000
TTY\$M-ST-QUOTING = 00400000
TTY\$M-ST-RDVERIFY = 00000400
TTY\$M-ST-READ = 00001000
TTY\$M-ST-RECALL = 00000800
TTY\$M-ST-SENDLF = 00000010
TTY\$M-ST-SKIPCRLF = 00080000
TTY\$M-ST-SKIPLF = 00002000
TTY\$M-ST-TABEXPAND = 00200000
TTY\$M-ST-TABRIGHT = 40000000
TTY\$M-ST-TYPFUL = 00001000
TTY\$M-ST-WRAP = 00008000
TTY\$M-ST-WRITE = 00000080
TTY\$M-ST-WRTALL = 00000010
TTY\$Q-RB-ECHOAREA = 0000000C
TTY\$READONE *****
TTY\$V-CH-LOWER = 00000003
TTY\$V-CH-SPEC = 00000004
TTY\$V-PC-NOCRLF = 00000007
TTY\$V-PC-NOTIME = 00000000
TTY\$V-RS-WRAP = 00000000
TTY\$V-ST-DEL = 00000001
TTY\$V-ST-ECHAES = 00000019
TTY\$V-ST-EDITING = 00000014
TTY\$V-ST-ESC O = 0000000E
TTY\$V-ST-OVERSTRIKE = 00000017
TTY\$V-ST-PRE = 0000001A
TTY\$V-ST-READ = 0000000C
TTY\$V-ST-SKIPLF = 0000000D
TTY\$V-SX-BACKSPACE = 00000005
TTY\$V-SX-BADESC = 00000028
TTY\$V-SX-CTRLO = 00000020
TTY\$V-SX-CTRLR = 00000032
TTY\$V-SX-CURSOR = 00000003
TTY\$V-SX-DEL = 00000021
TTY\$V-SX-EDITING = 00000034

x 02

TTY\$V-SX-EDITREAD = 00000009
TTY\$V-SX-EOL = 00000008
TTY\$V-SX-ESC = 00000027
TTY\$V-SX-ESCAPE = 0000002B
TTY\$V-SX-ESC O = 0000002E
TTY\$V-SX-FILC = 00000002
TTY\$V-SX-MULTI = 00000006
TTY\$V-SX-NINTMULTI = 00000038
TTY\$V-SX-NL = 00000029
TTY\$V-SX-NOECHO = 00000023
TTY\$V-SX-NOFLTR = 00000026
TTY\$V-SX-OVERSTRIKE = 00000037
TTY\$V-SX-OVRFLO = 00000030
TTY\$V-SX-PASALL = 00000022
TTY\$V-SX-PRE = 0000003A
TTY\$V-SX-QUOTING = 00000036
TTY\$V-SX-RDVERIFY = 0000000A
TTY\$V-SX-READ = 0000000C
TTY\$V-SX-RECALL = 0000000B
TTY\$V-SX-SENDLF = 00000004
TTY\$V-SX-SKIPCRLF = 00000033
TTY\$V-SX-SKIPLF = 0000002D
TTY\$V-SX-TABEXPAND = 00000035
TTY\$V-SX-TABRIGHT = 0000003E
TTY\$V-SX-TERMNORM = 00000038
TTY\$V-SX-TYPFUL = 0000002C
TTY\$V-SX-WRAP = 0000002F
TTY\$V-SX-WRITE = 00000007
TTY\$V-SX-WRTALL = 00000024
TTY\$WRITEDONE *****
TTY\$W-RB-AESLEN = 00000028
TTY\$W-RB-CPZCUR = 00000038
TTY\$W-RB-CPZORG = 0000003A
TTY\$W-RB-ESCTKN = 00000048
TTY\$W-RB-LINOFF = 00000030
TTY\$W-RB-LINREST = 00000032
TTY\$W-RB-MODE = 00000044
TTY\$W-RB-PRMLEN = 00000034
TTY\$W-RB-RDSTATE = 0000002A
TTY\$W-RB-TXTOFF = 0000003C
TTY\$W-RB-TXTSIZ = 00000040
TTY\$W-TA-INAHD = 0000000C
TTY\$W-TA-RCLOFF = 0000000E
TTY\$W-TA-RCLSIZ = 00000014
TTY\$W-WB-BCNT = 0000002A
TTY\$W-WB-STATUS = 00000028
TTY\$XON *****
TT-END = 00001583
UCB\$B-DEVTYPE = 00000041
UCB\$B-TT-CRFill = 000000F6
UCB\$B-TT-ESC = 00000103
UCB\$B-TT-ESC O = 00000104
UCB\$B-TT-FILC = 00000102
UCB\$B-TT-LASTC = 000000FF
UCB\$B-TT-LFFILL = 000000F7
UCB\$B-TT-LINE = 000000FE
UCB\$B-TT-OUTYPE = 0000010B

x 02

x 02
R 02

TT
VO

TTYCHARO
Symbol table

```
UCBSL_DEVDEPEND      = 00000044
UCBSL_DEVDEPN2       = 00000048
UCBSL_DUETIM         = 0000006C
UCBSL_IRP            = 00000058
UCBSL_SVAPTE         = 00000078
UCBSL_TT_MULT1       = 000000D8
UCBSL_TT_OUTADR      = 0000011C
UCBSL_TT_RDUE        = 000000B0
UCBSL_TT_SMLT        = 000000E0
UCBSL_TT_TYPAHD      = 000000E4
UCBSL_TT_WRTBUF      = 000000D4
UCBSM_INT            = 00000002
UCBSM_TIM            = 00000001
UCBSQ_TT_STATE       = 000000B8
UCBSV_INT            = 00000001
UCBSV_TT_TIMO        = 00000001
UCBSW_BCNT           = 0000007E
UCBSW_BOFF           = 0000007C
UCBSW_BUFQUO         = 00000018
UCBSW_DEVBUSIZ       = 00000042
UCBSW_DEVSTS         = 00000068
UCBSW_STS            = 00000064
UCBSW_TT_BSPLEN      = 00000100
UCBSW_TT_CURSOR      = 000000FC
UCBSW_TT_MULTILEN    = 000000DC
UCBSW_TT_OUTLEN      = 00000120
UCBSW_TT_PRTCTL      = 00000122
UCBSW_TT_SMLTLEN     = 000000DE
UPDATE_CURSOR        = 000010A9 R      02
UPDCURSOR            = 00000EE6 R      02
VERIFY_ARRAY         = ***** X 02
VTAB                  = 000003BE R      02
W0                    = 00000000
W1                    = 00000001
WRITE_END             = 00000093 R      02
WRITING               = 0000004D R      02
X                     = 00000000
X0                    = 00000000
X1                    = 00000001
XON                   = 00000632 R      02
Z0                    = 00000003
Z1                    = 00000001
ZERO_CPZORG          = 00001118 R      02
```

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$115_DRIVER	00001583 (5507.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	35	00:00:00.04	00:00:00.75
Command processing	117	00:00:00.35	00:00:02.68
Pass 1	880	00:00:31.65	00:01:18.64
Symbol table sort	0	00:00:03.01	00:00:05.46
Pass 2	533	00:00:08.06	00:00:16.13
Symbol table output	1	00:00:00.22	00:00:00.34
Psect synopsis output	0	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1568	00:00:43.34	00:01:44.01

The working set limit was 2400 pages.
273339 bytes (534 pages) of virtual memory were used to buffer the intermediate code.
There were 150 pages of symbol table space allocated to hold 2533 non-local and 289 local symbols.
3477 source lines were read in Pass 1, producing 29 object records in Pass 2.
52 pages of virtual memory were used to define 49 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-----	-----
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	20
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	29

2791 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:TTYCHARO/OBJ=OBJ\$:TTYCHARO MSRC\$:TTYCHARO/UPDATE=(ENH\$:TTYCHARO)+EXECMLS/LIB

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY